UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MODELANDO O PROBLEMA DA PRÓXIMA RELEASE SOB A PERSPECTIVA DA ANÁLISE DE PONTOS DE FUNÇÃO

Vitor Padilha Gonçalves

Orientador

Prof. Dr. Márcio de Oliveira Barros

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2014

Gonçalves, Vitor Padilha.
G635 Modelando o problema da próxima release sob a perspectiva da análise de pontos de função / Vitor Padilha Gonçalves, 2014.
xii, 81 f. ; 30 cm

Orientador: Márcio de Oliveira Barros.
Dissertação (Mestrado em Informática) - Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2014.

1. Engenharia de software. 2. Análise de pontos de função. 3. Heurística. 4. Problema da próxima release. I. Barros, Márcio de

MODELANDO O PROBLEMA DA PRÓXIMA RELEASE SOB A PERSPECTIVA DA ANÁLISE DE PONTOS DE FUNÇÃO

Vitor Padilha Gonçalves

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Prof. Márcio de Oliveira Barros, D.SC. - UNIRIO

Prof. Gleison dos Santos Souza, D. SC. - UNIRIO

Prof. Cláudia Maria Lima Werner, D. SC. - COPPE/UFRJ

RIO DE JANEIRO - RJ - BRASIL

SETEMBRO DE 2014

A minha mãe, aos meus irmãos, a minha avó, ao meu pai

Agradecimentos

Primeiramente, queria agradecer a DEUS que em todos os momentos de dificuldades recorri a ele para me apoiar e sempre fui atendido.

A minha mãe pelo carinho, amor e apoio acadêmico que me deu durante toda minha vida. Este apoio me permitiu alcançar este objetivo.

Aos meus irmãos Natália, Emília e Bruno, que com carinho e apoio me encorajaram a concluir este mestrado. A minha avó Leny e meu pai Waldemar que também contribuiu neste apoio.

Ao meu orientador Márcio Barros que esteve constantemente presente, desde o primeiro dia do mestrado, esclarecendo dúvidas, contribuindo com ideias e insumos necessários para conclusão da dissertação, e orientando como eu deveria conduzir a pesquisa.

A minha gerente Cláudia Fujie que permitiu conciliar minhas tarefas acadêmicas em concomitância com minhas tarefas profissionais.

A todos familiares e amigos, que compreenderam minha ausência nos momentos de estudo e de trabalho.

RESUMO

No contexto da Engenharia de Software, o problema de priorização de requisitos de softwares mais importantes para próxima *release*, chamado de Problema da Próxima Release (NRP), tem sido bastante discutido. Quanto ao número de objetivos, o problema é classificado em duas abordagens: (1) a primeira, mono-objetiva, tem como único objetivo a maximização da satisfação dos patrocinadores do projeto de software e estabelece uma restrição de investimento para a *release*; e (2) a segunda, bi-objetiva, em que a restrição de investimento é substituída por outro objetivo, que é minimizar o valor dos requisitos para a *release*. Tendo apenas um objetivo, a abordagem mono-objetiva gera apenas uma solução para o problema. A abordagem bi-objetiva é utilizada para encontrar as melhores soluções para diversas faixas de investimento e, sendo assim, várias soluções são apresentadas aos patrocinadores do projeto para tomada de decisão (escolher quais requisitos serão desenvolvidos na próxima *release*). No entanto, os trabalhos que tratam o problema para ambas as abordagens, apresentam formulações nas quais os requisitos possuem um custo fixo e indivisível. Na maioria destes trabalhos, os autores avaliam algoritmos no contexto do NRP com objetivo de avaliar a qualidade das soluções e o tempo de execução dos algoritmos.

Como a Análise de Pontos de Função (APF) é uma técnica não-linear para medição dos valores dos requisitos e é utilizada como base para estimativas de custos e prazos, sendo estas as principais variáveis para distribuição das *releases* de um projeto de software, este trabalho apresentará uma nova proposta de solução do NRP com base nesta técnica. Esta proposta pode ser formulada tanto de maneira mono-objetiva quanto bi-objetiva. De modo a explorar esta nova proposta, esta Dissertação apresentará dois estudos. O primeiro estudo tem como objetivo avaliar a formulação mono-objetiva, comparando-a com uma proposta mono-objetiva utilizada em trabalhos anteriores. Este estudo utiliza de instâncias (modelos de sistemas) reais e um algoritmo genético. O segundo estudo, utilizando as mesmas instâncias, avalia a qualidade das soluções geradas por três algoritmos (NSGA-II, SPEA2 e Aleatório) no contexto da formulação bi-objetiva, bem como o tempo de execução destes algoritmos.

Palavras-chave: Problema da Próxima Release, Pontos de Função, Heurísticas, Engenharia de Software Baseada em Buscas.

ABSTRACT

In Software Engineering, the problem of prioritizing the most important requirements for the next release of a software system, namely the *Next Release Problem* (NRP), has been widely discussed. According to the number of objectives, the problem is classified into two approaches: (1) the first, mono-objective, has the sole objective of maximizing the satisfaction of stakeholders, based on an investment restriction for a given release. (2) the second, bi-objective, on which the investment restriction is replaced by another objective - to minimize the value of the requirements for release. Having only one goal, the first approach generates a single solution. The bi-objective approach finds the best solutions to various investment groups and, therefore, several solutions are presented to stakeholders for decision-making. Works addressing the problem according to both approaches present formulations where software requirements have fixed, indivisible cost. In most cases, the authors evaluate algorithms in the context of the NRP by assessing the quality of produced solutions and the runtime of the selected algorithms.
Given that Function Point Analysis is a nonlinear technique for measuring the values of the requirements and is used as a basis for estimating development costs and schedule, which are the main variables for the distribution of releases of a software project, this paper presents a novel formulation for the NRP based on this technique. This proposal can be formulated as both a mono-objective and a bi-objective problem. In order to explore this new proposal, this Dissertation presents two studies. The first study evaluates the mono-objective formulation by comparing it to a classic mono-objective formulation used in previous works. It uses real system models as instances, along with a genetic algorithm. The second study, using the same instances, evaluates the quality of the solutions generated by three algorithms (NSGA-II, SPEA2 and random search) in the context of bi-objective formulation, as well as the runtime of these algorithms.

Keywords: Next Release Problem, Function Point, Heuristics, Search-based Software Engineering

Índice

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas Utilizadas

AG	Algoritmo Genético
APF	Análise de Pontos de Função
GRASP	Algoritmo Genético
HC	*Hill Climbing*
NRP	*Next Release Problem*
NRP-APF	*NRP* sob perspectiva da Análise de Pontos de Função
NRP-CLS	*NRP Clássico*
NSGA-II	*Non-Dominating Sorting Genetic Algorithm*
SA	*Simulated Annealing*
SPEA2	*Strength Pareto Evolutionary Algorithm*
PAES	*Pareto Archived Evolution Strategy*

Capítulo 1 – Introdução e Motivação

No contexto de contratação de serviços para desenvolvimento de software, usualmente o custo e o tempo necessários para entrega do produto são fatores relevantes. Estes fatores estão diretamente ligados ao esforço necessário para construção do sistema. Este esforço, por outro lado, é dependente dos requisitos desejados pelos patrocinadores do projeto. Muitas vezes, o esforço requerido para desenvolver um software é suficientemente grande para justificar seu particionamento em diversas versões (*releases*), de modo a atender às restrições financeiras e a data de entrega de alguns dos requisitos (DEL SAGRADO et al., 2011). Com o objetivo de aumentar a satisfação dos patrocinadores, os requisitos mais importantes devem ser priorizados. A escolha dos requisitos que serão incluídos em uma versão do software é comumente chamada de "problema da próxima release" (NRP, sigla em inglês para *next release problem*) (BAGNALL et al., 2001; DEL SAGRADO et al., 2011; DURILLO et al., 2011; ZHANG et al., 2013).

A impossibilidade de investigar todas as possíveis combinações de requisitos, mesmo em um problema de pequena escala, levou diversos autores a propor abordagens baseadas em buscas heurísticas para encontrar boas soluções para este problema (BAGNALL et al., 2001; DURILLO et al., 2011; LI et al., 2010; SOUZA et al., 2011; TONELLA et al., 2013). Estas soluções indicam um conjunto de requisitos que tenta maximizar a satisfação dos patrocinadores, enquanto atende a restrições de custo e\ou prazo. Nesta abordagem, o problema é caracterizado por ter um único objetivo (mono-objetivo) que é maximizar a satisfação dos patrocinadores do projeto de software. Neste sentido, a abordagem é apropriada quando se tem investimento máximo (seja de tempo, como de custo) pré-definido a ser empregado na *release*.

Como um exemplo de aplicação da abordagem mono-objetiva no contexto de desenvolvimento de software, considere uma empresa ABC que tem como meta gastar R$50.000,00 reais por *release* em um software SOFT com custo total de R$100.000,00 reais. Neste contexto, o objetivo é trazer para a primeira entrega os requisitos do

software mais importantes para os patrocinadores, podendo atender, por exemplo, 60% da expectativa destes com um investimento de apenas 50% do valor total do software.

No entanto, pode ser que esta mesma empresa não tenha uma meta de investimento por *release* (neste caso, não teremos a restrição), ainda que seja importante para ela definir quais requisitos serão desenvolvidos por *release*. Neste caso, o objetivo de maximizar a satisfação dos patrocinadores é alinhado com o objetivo de diminuir o custo dos requisitos para a *release*. Com isto, o conjunto de soluções ao problema é composto pelas soluções que apresentem a melhor satisfação para determinado valor de custo ou o menor custo para determinado nível de satisfação. Sendo assim, várias soluções são apresentadas e os patrocinadores devem escolher qual será adotada.

Como exemplo de aplicação da abordagem bi-objetiva ao NRP, considere que a empresa ABC quer saber, em um primeiro momento, qual é o escopo para primeira *release* do software SOFT. Duas soluções ao problema são apresentadas (podem existir várias outras): a primeira, com custo de R$ 50.000,00, atende a 60% das expectativas dos patrocinadores, enquanto a segunda, com custo de R$ 51.000,00, atende a 66% destas expectativas. Neste caso, pode ser que a empresa ABC adote a segunda solução, já que com apenas R$1.000 reais a mais ela eleva em 6% as expectativas dos patrocinadores.

Em grande parte das propostas ao NRP, seja mono-objetiva ou bi-objetiva, os autores atribuem um valor fixo ao esforço (ou custo) necessário para o desenvolvimento de cada requisito. Alguns trabalhos (DEL SAGRADO et al., 2011; ZHANG et al., 2013) consideram relações de interdependência entre requisitos, de modo que determinado requisito pode afetar e/ou ter seu esforço (ou custo) de desenvolvimento afetado pela presença (ou ausência) de outros requisitos na mesma versão do software. A seleção do subconjunto ideal de requisitos a ser implementado na próxima versão de um software é um problema NP-completo mesmo quando não existem interdependências entre os requisitos (BAGNALL et al., 2001).

Conforme o Guia Prático para Contratação de Soluções de Tecnologia da Informação (BRASIL, 2011) do Ministério do Planejamento, Orçamento e Gestão do Governo brasileiro, a contratação de serviços de TI por empresas e instituições do Governo depende de uma análise de custos e prazos apresentados em uma licitação. Como o esforço para desenvolvimento de um sistema é proporcional a estes dois fatores, a contratação de serviços de TI pelo Governo dependerá, em última análise, de uma estimativa do esforço. O esforço necessário para o desenvolvimento de um

software, em muitos casos, é estimado a partir do tamanho do software, que pode ser medido através de diversas técnicas (BOEHM et al., 2000; COSMIC, 2003; IFPUG, 2009), entre elas a análise pontos de função (APF) (AHN et al., 2003; FERREIRA; HAZAN, 2010; MATSON et al., 1994). O governo brasileiro, através de recomendação do TCU, tem utilizado a APF (IFPUG, 2009) como base para contratação de serviços de desenvolvimento de software para empresas e instituições públicas (FERREIRA; HAZAN, 2010). Uma vantagem de utilizar a APF é a flexibilidade de mudar o escopo ao longo do projeto: as empresas compram um volume de esforço para desenvolvimento de sistemas e não os sistemas propriamente ditos, permitindo mudanças nos requisitos e não ficando limitadas a um conjunto de necessidades iniciais.

Como mostrado nos parágrafos anteriores, existe a necessidade de particionamento dos requisitos de software em *releases* e, também, a aceitação da APF como métrica para medir o tamanho destes requisitos (derivando tempo e custo de desenvolvimento, por exemplo). Com base nisto, a principal contribuição desta dissertação é unir o NRP e a APF em uma mesma formulação, criando um método que pode ser utilizado no contexto desenvolvimento de software. Portanto, a motivação deste trabalho é a definição e avaliação de um método para particionamento do processo de desenvolvimento de software em releases aplicando uma técnica de medição de requisitos de software aceita pelo mercado.

1.1 Objetivo da Dissertação

Este trabalho aborda um problema recorrente na maioria das organizações que usualmente dividem a entrega de um software em *releases*. Nesse contexto, a revisão de literatura, que será apresentada no capítulo 2, mostra que os trabalhos que tratam do NRP não definem nenhuma técnica para medição dos requisitos do software. Com objetivo de explorar esta falta de definição, este trabalho aplica a APF na formalização do NRP e nomeia o novo método como o problema da próxima *release* sob a perspectiva da análise de pontos de função (NRP-APF). Este trabalho apresenta dois modelos formais para o problema, um para o NRP com investimento máximo desejado e um para tomada de decisão.

Ambos os modelos utilizam da mesma base em relação aos conceitos e restrições da APF. Porém, o modelo do NRP com investimento máximo desejado serve para quando há um limite a ser investido e, sendo assim, o modelo propõe encontrar uma única solução que atenda aos patrocinadores. O modelo para tomada de decisão

apresentará diversas boas soluções para que os patrocinadores escolham qual será utilizada para desenvolvimento do software.

Apresentada a formalização, buscas heurísticas são aplicadas aos modelos para que as soluções, ou uma única solução no caso do NRP com investimento máximo, sejam encontradas. Para avaliar a formalização proposta, dois estudos foram feitos do escopo desta dissertação. O primeiro estudo tem o objetivo de validar o NRP-APF, comparando-o com uma proposta de NRP bastante utilizada na literatura, que nomeamos NRP clássico e apresentaremos na seção 2. O segundo estudo se direciona em encontrar qual a melhor heurística a ser aplicada ao NRP-APF, ou seja, qual heurística produz melhores resultados quando utilizada para encontrar soluções para o NRP-APF.

1.2 Organização da Dissertação

Este trabalho está organizado em seis capítulos. O primeiro capítulo compreende esta introdução. O segundo capítulo, Buscas Heurísticas e o Problema da Próxima *Release*, faz um resumo sobre buscas heurísticas e faz uma revisão bibliográfica sobre trabalhos que tratam do problema da próxima *release*.

O terceiro capítulo, O Problema da Próxima *Release* sob a Perspectiva da Análise de Pontos de Função, formaliza o problema da próxima release utilizando a análise de pontos de função como técnica para medição do esforço de desenvolvimento dos requisitos do software.

O quarto capítulo, Avaliação do NRP baseado em Pontos de Função, descreve os dois experimentos conduzidos, quais as questões que nortearam a pesquisa, as instâncias utilizadas nos experimentos, quais algoritmos produzem soluções com melhor qualidade para o problema e faz uma análise dos resultados. Ao fim, discute as ameaças a validade do experimento como um todo e tira algumas conclusões.

Por fim, o sexto capítulo contém a conclusão do trabalho e considerações sobre os trabalhos futuros relacionados.

Capítulo 2 - Buscas Heurísticas e o Problema da Próxima *Release*

O problema da próxima *release* (*next release problem*, NRP, em inglês) consiste em priorizar requisitos de software de maneira a escolher um subconjunto de requisitos, a ser implementado na próxima versão do software, que maximize a satisfação dos patrocinadores e que atenda a algumas restrições. Algumas das restrições que influenciam na necessidade de priorizar os requisitos do software são o prazo, recursos (LI et al., 2010) e custo (BAGNALL et al., 2001). (BAGNALL et al., 2001) apresentam um dos primeiros trabalhos a tratar do assunto e introduziram o termo "*Next Release Problem*".

A escolha do subconjunto de requisitos de software pode ser feita de duas maneiras. A primeira é varrendo todos os subconjuntos possíveis de requisitos e verificando qual deles atende às restrições impostas e gera a maior satisfação dos patrocinadores. Sabendo que o total de subconjuntos de um conjunto é dado por 2^n, sendo *n* o número de requisitos candidatos à seleção, então o número de possíveis soluções do problema cresce exponencialmente ao número de requisitos candidatos. Em sistemas grandes, varrer todas as soluções possíveis pode ser inviável, já que o tempo necessário para percorrer todas as soluções e medir a satisfação de cada uma é muito alto. Em alguns casos, pode demorar desde dias até séculos para verificar todos os subconjuntos.

BAGNALL et al. (2001) demonstrou que o NRP é classificado como um problema NP-completo de otimização combinatória. A fim de não verificar todas as soluções, heurísticas são utilizadas para a escolha do subconjunto de requisitos a ser desenvolvido em determinada *release*. Heurísticas são técnicas que têm o objetivo de buscar boas soluções, próximas da solução ótima, para problemas combinatórios (REEVES, 1995).

Este capítulo apresenta trabalhos de pesquisa que tratam de mecanismos baseados em busca heurística para encontrar boas soluções para o NRP e as principais

diferenças entre estas abordagens. Entre estas diferenças, podemos citar: (1) os métodos utilizados para medir o esforço para desenvolver determinado requisito, bem como para relacionar seu grau de importância aos patrocinadores do sistema; (2) as dependências entre os requisitos; (3) o número de objetivos do problema, neste caso, sendo classificado como mono-objetivo e multi-objetivo; e (4) as heurísticas adequadas ao problema, de acordo com suas características.

A partir das diferentes abordagens do problema, dividiremos este capítulo em cinco seções. A primeira formaliza o NRP clássico, de acordo com BAGNALL et al. (2001). A segunda seção aborda os diferentes métodos de medição do valor dos requisitos e seu grau de importância para cada patrocinador, tendo em vista que os objetivos do NRP são minimizar o esforço para desenvolvimento dos requisitos que estarão na *release* e/ou maximizar a satisfação final dos patrocinadores. A terceira mostra as relações entre os requisitos, bem com os impactos relacionados às interdependências entre os requisitos no contexto do NRP. A quarta seção apresenta as diferentes abordagens quanto ao número de objetivos do problema. Neste contexto, alguns trabalhos consideram apenas um objetivo, que é maximizar a satisfação dos patrocinadores, tendo em vista que o esforço para desenvolvimento dos requisitos do subconjunto a ser escolhido é fixo e atribuído (BAGNALL et al., 2001; DEL SAGRADO et al., 2011; TONELLA et al., 2010). Em outros estudos, os objetivos são maximizar a satisfação do patrocinador e minimizar o custo total dos requisitos escolhidos. Neste sentido, não existe apenas uma boa solução, mas diversas boas soluções que são apresentadas aos patrocinadores de maneira que eles escolherão qual será a solução (DURILLO et al., 2011; ZHANG et al., 2013). A quinta seção mostra as heurísticas e/ou algoritmos aplicados ao NRP de acordo com as características a serem estudadas por diversos autores, bem como os resultados obtidos em estudos comparativos. E, por último, a seção de conclusão resumi os conceitos e trabalhos discutidos neste capítulo.

2.1 A Formulação Clássica do NRP

BAGNALL et al. (2001) formalizam o problema da NRP com único objetivo (mono-objetivo): maximizar a satisfação dos patrocinadores do projeto de software. Neste sentido, o esforço para desenvolvimento dos requisitos desejados para a próxima *release* possui um limite superior e é definido pelos patrocinadores de acordo com os recursos disponíveis na organização. Com isto, o objetivo é encontrar o subconjunto de

requisitos que melhor atende às expectativas dos patrocinadores. A seguir, sumarizamos a formalização do NRP de acordo com o trabalho de BAGNALL et al. (2001).

Considere um conjunto de requisitos $R = \{r_1, r_2, \dots, r_n\}$. Cada requisito r_i é descrito por um custo c_i, positivo e maior que zero, e possui um conjunto de precedências, que são os requisitos que devem ser implementados antes de r_i.

Considere $S = \{s_1, s_2, \dots, s_m\}$ como o conjunto de todos os patrocinadores do sistema. Cada patrocinador s_j possui um grau de interesse v_{ij} sobre cada requisito $r_i \in R$ e está associado a um peso p_j, que retrata a importância do patrocinador na empresa.

Neste sentido, é preciso encontrar um subconjunto de requisitos $R' \subseteq R$ que maximize a satisfação dos patrocinadores e onde o esforço para o desenvolvimento dos requisitos selecionados seja menor do que um limite de investimento representado por *B*.

$$\sum_{i \in R'} c_i \leq \mathrm{B} \quad (1)$$

$$\text{Maximizar } f(2) = \sum_{j=1}^{m} p_j \sum_{i \in R'} v_{ij} \quad (2)$$

Percebe-se que a equação (1) não é uma função objetivo, mas uma restrição imposta sobre possíveis soluções para o problema. Neste caso, somente a equação (2) é uma função objetivo do problema.

Por não haver qualquer restrição sobre a seleção de requisitos, o número de possíveis subconjuntos de requisitos $R' \subseteq R$ é 2^n, onde n é o número de requisitos disponíveis para a próxima *release* do software. Portanto, trata-se de um problema NP-completo, conforme comprovado em (BAGNALL et al., 2001).

2.2 Medição do Esforço e Satisfação dos Patrocinadores

As duas principais variáveis no contexto do NRP são o esforço para desenvolvimento dos requisitos e a satisfação dos patrocinadores. Na grande maioria dos trabalhos relacionados ao NRP, o esforço para desenvolvimento de um requisito é um valor constante e recebido como parâmetro definido nas instâncias do problema (BAGNALL et al., 2001; DURILLO et al., 2011; TONELLA et al., 2013; ZHANG et al., 2007).

Conforme visto na seção anterior, BAGNALL et al. (2001) representam como R o conjunto de todos os requisitos do software. Para cada requisito r_i, existe um custo representado por c_i. Na mesma linha, DEL SAGRADO et al. (2011) definem que para

cada requisito r_i existe e_i correspondente ao esforço para desenvolvimento do requisito. DURILLO et al. (2011) dizem que para cada requisito existe um custo para a sua satisfação. Neste sentido, o método utilizado para obter este valor é indiferente, ou seja, os autores não detalham como este valor deve ser atribuído ao requisito; apenas citam que cada requisito possui um esforço ou custo para desenvolvimento.

Para o cálculo da satisfação dos patrocinadores, alguns trabalhos atribuem, para cada requisito do software e cada patrocinador, um valor v_{ij} que representa o grau de importância do requisito *i* para o patrocinador *j* (BAGNALL et al., 2001; DURILLO et al., 2011; TONELLA et al., 2013). Em um sistema com *n* requisitos e *m* patrocinadores, existirão *n* x *m* atribuições de grau de importância. Com isto, é possível calcular a satisfação de cada patrocinador para o subconjunto de requisitos selecionado para a *release* como o somatório dos valores do grau de importância de cada requisito deste subconjunto para o patrocinador.

Desde os primeiros trabalhos sobre o NRP (BAGNALL et al., 2001), considera-se também o peso de cada patrocinador. Neste sentido, os requisitos que são de interesse de patrocinadores mais importantes terão um peso maior no momento da priorização e seleção para a próxima *release* (BAGNALL et al., 2001; DEL SAGRADO et al., 2011; DURILLO et al., 2011; ZHANG; HARMAN, 2010; ZHANG et al., 2013).

Em alguns projetos de software, várias pessoas podem estar envolvidas e considerar todas no momento da priorização pode ser inviável, pois levantar os interesses de cada um pode demandar muito tempo e esforço. Neste sentido, LIM et al. (2012) propõem um método que se utiliza de redes sociais e de um algoritmo genético para encontrar patrocinadores "chave" no projeto de software. Este método pode ajudar a selecionar quais patrocinadores deveriam ser considerados no momento da priorização.

2.3 Dependências entre Requisitos do Software

As dependências entre os requisitos refletem como os requisitos do software interagem entre eles (ZHANG et al., 2013). Esta interação pode gerar implicações nos requisitos envolvidos nestas relações de dependências, tais como, aumento no custo dos requisitos envolvidos, alteração no grau de interesse por parte dos patrocinadores associados a estes requisitos, entre outros. A importância de se considerar estas dependências de requisitos de software no NRP é tratada desde o trabalho de BAGNALL et al. (2001).

Apesar disto, alguns trabalhos não consideram estas dependências ao formalizar o NRP e aplicar buscas heurísticas para resolvê-lo. DURILLO et al. (2010) e ZHANG et al. (2007) apenas consideram os custos dos requisitos e seu grau de importância para os patrocinadores. Neste sentido, o subconjunto de requisitos a ser considerado em uma *release* pode ser escolhido sem considerar as restrições e impactos impostos pelas dependências.

BAGNALL et al. (2001) tratam apenas a dependência relativa à precedência entre requisitos, ou seja, um requisito r_i só pode ser desenvolvido depois de um subconjunto de requisitos que o precede, representado por $prec(r_i)$. Desta maneira, estas dependências podem ser representadas por um grafo direcional acíclico, como representado pelos quadrados e setas direcionais da Figura 1. Considerando as restrições, o conjunto de possíveis soluções diminui. Porém, o NRP continua sendo um problema NP-completo, pois mesmo as instâncias com dependências podem ser formuladas como o NRP básico, dado que cada requisito r_i pode ser reconsiderado como $r_i \cup prec(r_i)$ (BAGNALL et al., 2001). Da mesma maneira que BAGNALL et al. (2001), DURILLO et al. (2009), JIANG et al. (2010) e TONELLA et al. (2013) tratam apenas a dependência de precedência entre requisitos e utilizam o grafo direcional acíclico para representar estas relações.

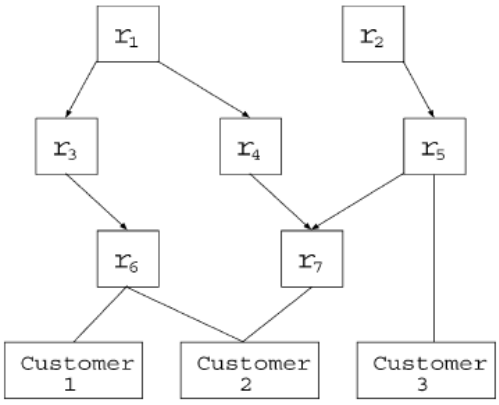

Figura 1. Um exemplo de estrutura do NRP (BAGNALL et al., 2001)

GREER e RUHE (2004) relatam, além das precedências, a relação AND, onde um requisito r_1 não pode ser selecionado separadamente de um requisito r_2. DEL SAGRADO et al. (2011) tratam, além das duas anteriores (precedência e AND), mais três tipos de dependências. A primeira dependência, denominada *Exclusão*, acontece quando um requisito r_1 não pode ser selecionado juntamente com um requisito r_2. A segunda dependência, chamada de *Cost-based* (*Cost-Related* por ZHANG et al. (2011)), ocorre quando um requisito r_1 altera o esforço de desenvolvimento de um requisito r_2. E

a terceira dependência, denominada *Revenue-based* (*Value-Related* por ZHANG et al. (2011)), ocorre quando um requisito r_1 interfere no grau de interesse dos patrocinadores do projeto sobre um requisito r_2. Neste sentido, além das dependências do trabalho de DEL SAGRADO et al. (2011), mais duas relações são descritas. No trabalho de ZHANG et al. (2011) também são considerados estes cinco tipos de dependências. A Tabela 1 apresenta estas relações e suas definições de acordo com ZHANG et al. (2011). Das relações apresentadas na Tabela 1, todas podem existir simultaneamente entre dois requisitos (r_1 e r_2, por exemplo), exceto a *AND* juntamente com a *OR* e a *AND* com a precedência (ZHANG; HARMAN, 2010).

Tabela 1. Dependências entre Requisitos de Software no NRP (ZHANG et al., 2013)

AND	Quando um requisito r_1 é selecionado, o requisito r_2 também é escolhido.
OR	Os requisitos r_1 e r_2 são conflitantes entre si. Somente um entre r_1 e r_2 pode ser selecionado.
Precedência	O requisito r_1 é selecionado antes do requisito r_2.
Value-related	Quando r_1 é selecionado, esta seleção afeta o grau de importância de um requisito r_2 para um conjunto de patrocinadores.
Cost-Related	Quando r_1 é selecionado, esta seleção afeta o esforço para desenvolvimento de um requisito r_2.

LI et al. (2010) cita uma sexta dependência, denominada *Time-Related*, que é uma extensão da precedência. Esta dependência define um tempo no qual pode ser iniciado o desenvolvimento de um requisito a partir de outro. Um exemplo seria o desenvolvimento de um requisito r_1 que possa ser iniciado cinco dias após o início de r_2, mesmo que o tempo total para o desenvolvimento de r_2 seja de dez dias.

2.4 Mono-objetivo x Bi-objetivo

BAGNALL et al. (2001) formalizam o problema da NRP com único objetivo (mono-objetivo): maximizar a satisfação dos patrocinadores do projeto de software. Neste sentido, o esforço para desenvolvimento do software possui um limite superior e é definido pelos patrocinadores, conforme mostrado na Seção 2.2. GREER e RUHE (2004), LI et al. (2007) e FREITAS et al. (2011) também tratam o problema com este único objetivo, fixando o custo máximo por *release*.

SOUZA et al. (2011) também considera os riscos relacionados ao desenvolvimento dos requisitos. Apesar de considerar estes riscos, o trabalho possui

somente um objetivo, já que o risco e a importância dos requisitos são combinados em uma única função objetivo. Neste sentido, para formalização do problema, os autores consideram R = {r_1, r_2, ..., r_n} como o conjunto de *n* requisitos do software, C = {c_1, c_2, ..., c_m} como o conjunto dos *m* patrocinadores, P como o número de *releases*, $risco_i$ e $custo_i$ como o risco e custo associados a determinado requisito r_i. Com isto, a função objetivo é:

$$Maximizar \sum_{i=1}^{n} (score_i \cdot (P - x_i + 1) - risco_i \cdot x_i) \cdot y_i \quad (3)$$

sendo

$$score_i = \sum_{j=1}^{m} w_j \cdot importancia(c_j, r_i) \quad (4)$$

com restrição

$$\sum_{j=1}^{n} custo_i \cdot f_{i,k} \leq Investimento\ para\ Release, \forall\ k \in \{1, \ldots, P\}$$
$$x_b \leq x_a, \forall (x_a \text{ precedente a } x_b) \text{ e } x_a, x_b \in R \quad (5)$$

onde $y_i \in \{0, 1\}$ e indica se o requisito r_i será implementado em alguma *release*, x_i indica o número da *release* na qual r_i será implementado, $f_{i,k} \in \{0, 1\}$ indica se r_i foi implementado na release *k* e w_i representa a importância do patrocinador *j* no projeto de software.

O primeiro estudo a utilizar uma abordagem multi-objetivo para o NRP foi apresentado por ZHANG et al. (2007). Em um problema de otimização multi-objetivo, deseja-se encontrar um conjunto de valores para as variáveis de decisão que otimize um conjunto de funções objetivo. Como definido por ZHANG et al. (2007), um vetor de variáveis de decisão $\vec{x}$ é dominante em relação ao vetor $\vec{y}$ se:

$$f_i(\vec{x}) \geq f_i(\vec{y})\ \forall\ i\ \in \{1, 2, \ldots, M\} \quad (6)$$

e

$$\exists i \in \{1,2, \ldots, M\} | f_i(\vec{x}) > f_i(\vec{y}) \quad (7)$$

onde M é o número de objetivos do problema.

Todos os vetores de variáveis de decisão que não são dominados por nenhum outro são chamados de *não-dominados* ou "ótimo de Pareto" e formam a fronteira de Pareto (ZHANG et al., 2007). Uma fronteira de Pareto ótima é aquela que possui as melhores soluções em todo espaço de busca, ou seja, ela possui todas as soluções *não-dominadas* quando verificado todo espaço de busca. Estes vetores são as soluções que não podem ser melhoradas e, portanto, são os valores que deverão ser considerados como boas soluções. A Figura 2 ilustra um exemplo de fronteira de Pareto bi-objetiva.

Figura 2. Exemplo de Fronteira de Pareto para o NRP

Os diversos autores que tratam do NRP sob uma perspectiva multi-objetivo (DURILLO et al., 2011; ZHANG; HARMAN, 2010; ZHANG et al., 2007, 2013) utilizam duas funções objetivo, que são maximizar a satisfação dos patrocinadores e minimizar o esforço necessário para o desenvolvimento dos requisitos. Neste caso, *M* = 2 de acordo com a formalização apresentada anteriormente. É um problema de otimização também denominado de bi-objetivo (DURILLO et al., 2011). Considerando as variáveis já apresentadas na formalização do NRP mono-objetivo da Seção 2.1, as duas funções objetivo do NRP multi-objetivo, de acordo com ZHANG et al. (2007) e DURILLO et al. (2010), são:

$$\text{Minimizar } f(1) = \sum_{i \in R'} c_i \tag{8}$$

$$\text{Maximizar } f(2) = \sum_{j=1}^{m} p_j \sum_{i \in R'} v_{ij} \tag{9}$$

2.5 Algoritmos Aplicados ao NRP

Esta seção apresenta as heurísticas e/ou algoritmos encontrados na literatura que são aplicados ao NRP. PITANGUEIRA et al. (2013) realizaram uma revisão sistemática que considerou trinta trabalhos relacionados ao NRP e listou 32 heurísticas utilizadas na resolução do problema, conforme mostra a Figura 3. Na figura, o eixo y representa a quantidade de ocorrências do algoritmo e o eixo x os algoritmos utilizados.

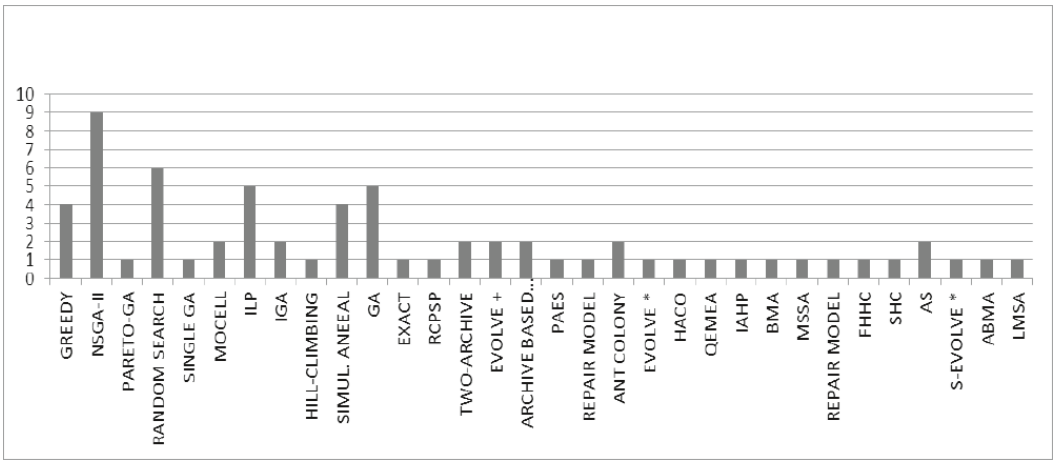

Figura 3. Heurísticas utilizadas no contexto do NRP (PITANGUEIRA et al., 2013)

A escolha das heurísticas adequadas ao problema depende do número de objetivos considerados na sua formalização. Neste sentido, esta seção é dividida em duas. A primeira subseção mostra as heurísticas mono-objetivo aplicadas ao NRP, bem como os resultados comparativos entre elas. A segunda subseção apresenta as heurísticas multi-objetivo aplicadas ao NRP.

2.5.1 Algoritmos para o NRP Mono-Objetivo

Algoritmos mono-objetivo são aqueles que consideram apenas um objetivo, ou seja, conseguem encontrar um bom resultado referente a apenas uma variável do problema. Diversas heurísticas para NRP mono-objetivo são relatadas na literatura, tais como *hill climbing* (BAGNALL et al., 2001; DEL SAGRADO; ÁGUILA, 2009; FREITAS et al., 2011), algoritmos genéticos (AG) (DEL SAGRADO et al., 2011; FREITAS et al., 2011; SOUZA et al., 2011), *tabu search* (SOUZA et al., 2011), *simulated annealing* (BAGNALL et al., 2001; DEL SAGRADO et al., 2011; FREITAS et al., 2011; SOUZA et al., 2011), *GRASP* (BAGNALL et al., 2001; SOUZA et al., 2011), colônia de formigas (DEL SAGRADO; ÁGUILA, 2009; JIANG et al., 2010; SOUZA et al., 2011), entre outros. No contexto do NRP, os autores utilizam estas heurísticas com objetivo de encontrar o subconjunto de requisitos que maximize a satisfação dos patrocinadores.

BAGNALL et al. (2001) avalia quatro algoritmos como solução para o NRP mono-objetivo. Os algoritmos escolhidos pelos autores foram um algoritmo de otimização exata (programação linear, no caso), o GRASP, o *Hill Climbing* e o *Simulated Annealing*. Os autores utilizaram cinco modelos de sistemas para avaliação dos algoritmos e executaram 10 vezes cada algoritmo para cada um dos cinco modelos. Os modelos têm diferentes tamanhos quanto ao número de requisitos e patrocinadores,

sendo que o menor possui 100 patrocinadores e 140 requisitos e o maior tem 500 patrocinadores e 3250 requisitos. Após análise da melhor solução encontrada por cada algoritmo, os autores concluem que para sistemas que envolvem poucos requisitos e patrocinadores, um algoritmo de otimização exata é suficiente. No entanto, para problemas maiores, o *Simulated Annealing* se mostrou mais eficiente ao encontrar os melhores resultados e em um tempo de execução satisfatório (BAGNALL et al., 2001).

FREITAS et al. (2011) propõem a utilização do método exato *simplex* para encontrar boas soluções para o NRP mono-objetivo. O método simplex é baseado na representação geométrica do problema na forma de equações lineares. Este método usa a formulação do problema em uma matriz e aplica operações matemáticas de matrizes para encontrar a melhor solução para o problema. Para avaliação do método, os autores confrontaram as médias dos resultados, quanto à satisfação dos patrocinadores e tempo de execução, obtidas pelo método e por duas heurísticas (algoritmo genético e *Simulated Annealing*). Nesta avaliação, foram utilizados cinco modelos de sistemas, sendo o menor com 20 requisitos e 10 patrocinadores e o maior com 200 requisitos e 100 patrocinadores. Em cada sistema, foram utilizados limites de orçamento de 30%, 50% e 70% do valor total do projeto. Os algoritmos heurísticos foram executados 100 vezes cada e o método exato foi executado uma só vez, pois retorna sempre a mesma solução. Após a comparação, o método exato apresentou resultados melhores em todos os cenários (total de 15 cenários: cinco modelos x três limites de orçamento) quando comparado aos outros dois algoritmos. Em relação ao tempo de execução, o método exato se mostrou mais rápido para os dois modelos maiores. Porém, as heurísticas encontraram a solução em menor tempo para os três menores modelos.

DEL SAGRADO et al. (2011) avaliaram os algoritmos GRASP, genético e colônia de formigas utilizando dois modelos de projetos de software. O primeiro modelo é o mesmo utilizado por GREER e RUHE (2004), que possui 20 requisitos, cinco patrocinadores e investimento máximo de 25 unidades de medida de esforço para os requisitos a serem considerados na *release*. Esta unidade de medida de esforço está associada a cada requisito e varia de 1 até 10, de acordo com o esforço estimado para o desenvolvimento do requisito. Os autores não definem um método para obter esta unidade de medida de esforço. O segundo é um modelo sintético com 100 requisitos, cinco patrocinadores e investimento máximo de 20. Após 100 execuções para cada projeto, a análise dos resultados possibilitou concluir que o GRASP produz melhores resultados em menor tempo de execução quando comparado aos demais.

Algumas heurísticas conhecidas, como o algoritmo genético, podem apresentar diferentes variações em sua implementação. TONELLA et al. (2010) propõem o uso do algoritmo genético interativo no contexto do NRP, de maneira a utilizar entradas de dados feitas pelos usuários para que o algoritmo siga um caminho para uma boa solução quando as informações iniciais são insuficientes. Desta forma, o usuário orienta o algoritmo a seguir por um bom caminho. Para avaliação desta proposta, os autores compararam o algoritmo interativo com um algoritmo genético que não recebe as informações ao longo do processo de otimização e uma busca aleatória em um sistema com 49 requisitos. Após 30 execuções de cada algoritmo, utilizando o teste ANOVA (apesar de não citar o nível de significância e se os dados eram normais) e análise de gráficos box-plot, os autores verificaram que o algoritmo genético interativo encontra melhores soluções quando comparado ao algoritmo genético básico e à busca aleatória.

SOUZA et al. (2011) propuseram o uso da heurística de colônia de formiga, que tenta "imitar" o comportamento das colônias de formigas em busca de comida na natureza, como solução ao NRP mono-objetivo com dependências de requisitos. O objetivo do estudo foi avaliar a qualidade (importância dos requisitos aos patrocinadores e risco) dos resultados, bem como o tempo de execução do algoritmo. Apesar dos autores utilizarem a importância dos requisitos e o risco como parâmetros para medição da qualidade dos resultados, estes dois fatores são agrupados e medidos em uma única função objetivo (representada pela função 3 da Seção 2.4). Para isto, o algoritmo que implementa a heurística de colônia de formigas foi comparado com dois outros algoritmos (Algoritmo genético e *Simulated Annealing*). Para avaliação da proposta, SOUZA et al. (2011) utilizaram 72 modelos de sistemas gerados sinteticamente, que apresentam diferentes configurações quanto à quantidade de requisitos, releases, patrocinadores e precedências, além do custo máximo por release. Após a execução dos três algoritmos sobre os modelos, o trabalho conclui que: (1) em todos os modelos, o algoritmo de colônia de formigas produziu resultados com melhor qualidade; (2) o algoritmo de colônia de formigas produziu, em média, um resultado 78,27% melhor que o algoritmo genético e 96,77% melhor que o *Simulated Annealing*; e (3) o algoritmo de colônia de formigas exige um tempo maior de execução se comparado aos outros dois algoritmos.

JIANG et al. (2010) avaliam a utilização de métodos híbridos - *Hill Climbing* (HC) com colônia de formigas – como proposta de solução ao NRP. Neste trabalho, os autores citam ter utilizado modelos de cinco sistemas distintos, gerados

automaticamente com os mesmos parâmetros usados nos modelos sintéticos apresentados por BAGNALL et al. (2001). Embora utilize os mesmos parâmetros para geração dos dados, os modelos de sistemas não são os mesmos. Neste sentido, para cada sistema o estudo utiliza três limites de orçamento baseados no custo total do sistema, sendo eles 30%, 50% e 70%, gerando um total de quinze (cinco modelos x três limites) instâncias a serem analisadas. Comparando os resultados de uma execução de cada algoritmo sobre estas amostras, o algoritmo proposto obteve os melhores resultados quanto à satisfação dos patrocinadores quando comparado com o GRASP, *Simulated Annealing*, *Hill Climbing* e colônia de formigas, apesar do tempo de execução ser superior a pelo menos um dos demais algoritmos em 11 das 15 instâncias.

2.5.2 Algoritmos para o NRP Multi-Objetivo

Como primeiro trabalho a abordar o NRP multi-objetivo, ZHANG et al. (2007) avaliam o uso de quatro algoritmos aplicados à formalização do problema. São eles: (1) o NSGA-II (DEB et al., 2002), que é uma variação do algoritmo genético simples que trata mais de um objetivo; (2) o algoritmo genético de Pareto (ZHANG et al., 2007); (3) o algoritmo genético de único objetivo; e (4) um algoritmo de busca aleatória. Sabendo que o algoritmo genético de único objetivo é usado em problemas com apenas um objetivo, os autores utilizaram um sistema de pesos (representado por ω) que transforma as duas funções objetivo do problema (representadas por $f_1(\vec{x})$ e $f_2(\vec{x})$) em uma única função objetivo. Esta função é dada por:

$$F(x) = (1 - \omega) \cdot f_1(\vec{x}) + \omega \cdot f_2(\vec{x}), \text{ onde } \omega \in [0,1] \quad (10)$$

Em um primeiro estudo experimental, baseado em estatística descritiva e análise de gráficos, ZHANG et al. (2007) utilizaram três modelos de software (15 patrocinadores e 40 requisitos; 50 patrocinadores e 80 requisitos; 100 patrocinadores e 140 requisitos) e executaram cada modelo cinco vezes por algoritmo. Após a análise dos gráficos com os resultados das execuções, os autores concluíram que: (1) o NSGA-II encontra os melhores resultados se comparado aos outros três algoritmos; (2) somente o algoritmo genético mono-objetivo encontrou soluções nos extremos de cada função objetivo, quando ω = 1 e ω = 0. No caso dos outros algoritmos, os autores propõem incluir estes resultados na lista das boas soluções (fronteira de Pareto), principalmente ao NSGA-II que obteve melhores resultados; e (3) quanto maior a escala do problema (número de patrocinadores e requisitos), maior é a diferença de desempenho entre os

algoritmos, ou seja, mais evidente fica que os resultados produzidos pelo NSGA-II são melhores que os demais.

No mesmo artigo, ZHANG et al. (2007) estudam o comportamento dos quatro algoritmos em situações extremas. Para isto, os autores usam mais três modelos de sistemas, as mesmas configurações de execução e análise de gráficos como forma de análise dos resultados. O primeiro modelo possui 100 patrocinadores e 20 requisitos; o segundo, 100 patrocinadores e 25 requisitos; e o terceiro, 2 patrocinadores e 100 requisitos. Na análise dos resultados para o primeiro modelo, os autores verificaram que não existe grande divergência entre as soluções geradas por todos os algoritmos. Para o segundo modelo, verifica-se o NSGA-II produz melhores soluções em relação aos demais, apesar da pouca variação entre os modelos, já que o segundo modelo só possui cinco requisitos a mais. E, por último, a análise do terceiro modelo mostrou que o NSGA-II também produz melhores resultados em relação aos outros três algoritmos.

DURILLO et al. (2009) comparam o NSGA-II, o MOCell - também uma variação do algoritmo genético simples adaptado ao contexto multi-objetivo e introduzido por NEBRO et al. (2009) – e um algoritmo aleatório. Neste trabalho, os autores utilizaram dois indicadores de qualidade para avaliar a qualidade dos resultados obtidos por estes três algoritmos: *spread* e *hipervolume*. O *spread* mede a diversidade atribuída à distribuição dos valores encontrados ao longo da fronteira de Pareto gerada por cada algoritmo. O *hipervolume* é um indicador que mede o volume do espaço de busca gerado pelas soluções não-dominadas (fronteira de Pareto) para problemas em que todos os objetivos são minimizados (DURILLO et al., 2009). Uma fronteira de Pareto com maior *hipervolume* do que outra pode significar que (a) algumas soluções desta fronteira de Pareto dominam soluções da segunda; e (b) suas soluções são melhor distribuídas do que as soluções da segunda. Sabendo que convergência é uma medida que diz respeito à distância das soluções da fronteira de Pareto encontrada para as soluções da fronteira de Pareto ótima – quanto menor a distância, maior a convergência -, o *hipervolume* é um indicador de convergência e diversidade da fronteira encontrada pelo algoritmo. Segundo autores, a qualidade do resultado é inversamente proporcional ao *spread* e proporcional ao *hipervolume*. Estes indicadores serão apresentados novamente, porém com mais detalhes, no Capítulo 4. Utilizando seis modelos de sistemas que não consideram as dependências entre os requisitos, os três algoritmos foram executados para avaliar os dois indicadores de qualidade. Os autores concluíram

que o algoritmo MOCell que produziu, em média, resultados com menor *spread* e o NSGA-II com maior *hipervolume*.

DURILLO et al. (2010) avaliam, além dos algoritmos MOCell e NSGA-II, um algoritmo proposto por KNOWLES E CORNE (1999) denominado PAES. No estudo comparativo entre o PAES, MOCell e NSGA-II, os autores utilizaram seis modelos de sistemas (as mesmas utilizadas por ZHANG et al. (2007)), executando os modelos 100 vezes em cada algoritmo, com configuração de 5.000, 10.000 e 25.000 avaliações máximas. O objetivo do estudo foi comparar estes algoritmos quanto ao *hipervolume*, *spread*, número de soluções não dominadas encontradas e tempo de execução para cada faixa de avaliação.

DURILLO et al. (2010) utilizaram o teste Kolmogorov–Smirnov para teste de normalidade, o teste ANOVA caso os dados apresentassem distribuição normal e o teste não paramétrico Kruskal–Wallis, caso contrário. Com nível de confiança igual à 95%, os testes do estudo afirmam que: (1) com relação ao *hipervolume*, o NSGA-II encontra melhores soluções que o MOCell na maioria dos modelos de sistema e nas diferentes faixas, enquanto o PAES apresenta os piores resultados; (2) quanto ao *spread*, o MOCell apresenta as melhores médias em todas as faixas e modelos, enquanto o PAES, novamente, apresenta os piores resultados em todas as execuções; (3) em relação ao números de soluções não dominadas encontradas, para um número de avaliação máxima menor (5.000), o NSGA-II encontrou um maior número de soluções e, a medida que se aumentava o número máximo de avaliações permitidas, o MOCell igualou-se ao NSGA-II, enquanto o PAES não apresentou, em nenhum caso, um resultado melhor que os demais algoritmos; (4) quanto ao tempo de execução, o PAES apresentou um tempo menor na maioria das execuções e, somente para um modelo com mais requisitos e patrocinadores, o NSGA-II foi um pouco mais rápido que o PAES.

2.6 Considerações Finais

Conforme mostrado neste capítulo, o NRP é comumente caracterizado pelo esforço de desenvolvimento exigido pelos requisitos, pelo grau de importância destes requisitos para os patrocinadores do projeto e pelas dependências entre os requisitos. Na maioria dos trabalhos citados neste capítulo, o esforço para desenvolvimento e o grau de importância dos requisitos são definidos por valores constantes e informados pelo usuário como parâmetro do problema. No entanto, estes trabalhos não discutem os métodos que podem ser utilizados para definir estes valores. Mais ainda, estes valores

podem ser influenciados pelas dependências entre os requisitos. As dependências entre os requisitos, por sua vez, também são responsáveis também por definir a ordem em que eles podem ser desenvolvidos. Além disto, o número de objetivos do problema vai definir a quantidade de boas soluções que serão apresentadas aos patrocinadores, com o mono-objetivo apresentando apenas uma boa solução e o bi-objetivo um conjunto de boas soluções (fronteira de Pareto). A Tabela 2 sumariza os trabalhos relacionados quanto aos algoritmos utilizados, as funções objetivo e as restrições consideradas no problema.

Tabela 2. Resumo dos Trabalhos Relacionados

Referência	Algoritmos utilizados	Funções objetivo consideradas	Restrições consideradas
BAGNALL et al. (2001)	Técnica Exata GRASP *Hill Climbing (HC)* *Simulated Annealing*	Satisfação dos Patrocinadores	Precedência Custo
GREER e RUHE (2004)	Algoritmo Genético (AG)	Satisfação dos Patrocinadores	Precedência Dependência AND Esforço
ZHANG et al. (2007)	NSGA-II AG de Pareto AG Mono-objetivo Busca Aleatória	Satisfação dos Patrocinadores	Custo
DURILLO et al. (2009)	NSGA-II MOCell	Custo Satisfação dos Patrocinadores	Precedência
DURILLO et al. (2010)	NSGA-II MOCell PAES	Custo Satisfação dos Patrocinadores	N\A
JIANG et al. (2010)	GRASP *Hill Climbing (HC)* *Simulated Annealing* *Colônia de Formigas (CF)* *Híbrido de HC com CF*	Satisfação dos Patrocinadores	Precedência Custo
TONELLA et al. (2010)	AG Interativo AG (Não Interativo) Busca Aleatória	Divergências nas relações (Penalidade para violação de restrições)	Precedência Prioridade
DEL SAGRADO et al. (2011)	GRASP AG Colônia de Formiga	Satisfação dos Patrocinadores	Precedência Dependência AND Dependência OR Dependência de custo Dependência de valor Custo
FREITAS et al. (2011)	Técnica Exata *Simulated Annealing* AG Busca Aleatória	Satisfação dos Patrocinadores	Precedência Custo

LI et al. (2010)	Técnica Exata	Tempo do Projeto	Precedência Dependência AND Dependência OR Dependência de Custo Dependência de Valor Dependência de Tempo
SOUZA et al. (2011)	AG *Simulated Annealing* Colônia de Formiga	Satisfação dos Patrocinadores e Risco (Mono-objetivo)	Precedência Custo
ZHANG et al. (2011)	NSGA-II	Custo Satisfação dos Patrocinadores	Precedência Dependência AND Dependência OR Dependência de Custo Dependência de Valor
ZHANG et al. (2013)	NSGA-II	Custo Satisfação dos Patrocinadores	Precedência Dependência AND Dependência OR Dependência de Custo Dependência de Valor

O próximo capítulo formalizará o NRP sob a perspectiva da Análise de Pontos por Função tomando como base os conceitos apresentados neste capítulo.

Capítulo 3 – O Problema da Próxima *Release* sob a Perspectiva da Análise de Pontos de Função

Os estudos que apresentam soluções para o problema da próxima *release*, como discutido no capítulo anterior, não o relacionam com nenhuma técnica que permita medir o esforço necessário para desenvolvimento dos requisitos do software. Como vimos no Capitulo 2, alguns trabalhos consideram relações de interdependência entre requisitos (DEL SAGRADO et al., 2011; ZHANG et al., 2013), de modo que determinado requisito pode afetar e/ou ter seu esforço (ou custo) de desenvolvimento afetado pela presença (ou ausência) de outros requisitos na mesma versão do software. Porém, nestes estudos, os esforços (ou custos) para desenvolvimento de cada requisito são valores fixos, pré-determinados e informados pelo responsável pelo planejamento da *release*.

No contexto da Engenharia de Software, diversas técnicas podem ser utilizadas para medir e/ou estimar o esforço de desenvolvimento de software (BOEHM et al., 2000; COSMIC, 2003; IFPUG, 2009), entre elas a Análise de Pontos de Função (APF) (AHN et al., 2003; FERREIRA; HAZAN, 2010; MATSON et al., 1994). Neste trabalho, a APF foi escolhida como técnica para medição de esforço no contexto do NRP. A motivação para esta escolha é a grande aceitação da APF no mercado de software, inclusive pelo governo brasileiro que, através de recomendação do TCU, tem utilizado a APF (IFPUG, 2009) como base para contratação de serviços de desenvolvimento de software para empresas e instituições públicas (FERREIRA; HAZAN, 2010). Utilizando a APF, poderemos calcular o esforço necessário para o desenvolvimento de uma *release* a partir de informações que compõem a descrição dos requisitos selecionados, explorando não apenas a presença ou ausência de um determinado requisito em uma *release*, mas a presença ou ausência de partes deste requisito, que podem reduzir o esforço necessário para sua implementação na *release*, assim como reduzir o custo de outros requisitos.

Neste sentido, este capítulo apresenta o NRP sob a perspectiva da APF e está dividido em cinco seções: a primeira mostra conceitos da APF; a segunda seção trata a formalização do NRP sob a perspectiva da APF, tomando como base o trabalho de GONÇALVES e BARROS (2013); a terceira apresenta uma proposta de solução para o problema; por fim, a quarta conclui a respeito da formalização e da proposta de solução discutida nas seções anteriores.

3.1 Análise de Pontos de Função

A contagem de pontos de função consiste em contar funções de dados e de transações. Uma função de dados é uma "funcionalidade de dados que satisfaz os Requisitos Funcionais do Usuário referentes a armazenar e/ou referenciar dados", enquanto uma função de transação é um "processo elementar que fornece funcionalidade ao usuário para processar dados" (IFPUG, 2009). Neste sentido, a existência de uma função de dados é decorrente das necessidades de armazenamento e recuperação de dados para atender ao processamento realizado nas funções de transação. Só faz sentido uma função de dados existir se ela for utilizada por alguma função de transação. Caso contrário, estaríamos contabilizando um elemento que não possui nenhuma utilidade aos usuários do software.

As funções de dados podem ser classificadas como ALI (arquivo lógico interno) ou AIE (arquivo de interface externa). Um ALI é definido como um "grupo de dados ou informações de controle logicamente relacionados, reconhecido pelo usuário, mantido dentro da fronteira da aplicação", ou seja, refere-se a informações que podem ser incluídas ou alteradas pelo próprio sistema. Já um AIE é um "grupo de dados ou informações de controle logicamente relacionados, reconhecido pelo usuário e referenciado pela aplicação, porém mantido dentro da fronteira de outra aplicação", ou seja, estas informações são somente consultadas pelo sistema a ser desenvolvido.

Cada função de dados é caracterizada por dois elementos: dados elementares referenciados (DER) e registros lógicos referenciados (RLR). Segundo o IFPUG, um DER é um "atributo único, reconhecido pelo usuário e não repetido" e um RLR é "um subgrupo de dados elementares referenciados, reconhecido pelo usuário e contido em uma função de dados". Ou seja, um DER está contido em um RLR, que está contido em uma função de dados. A complexidade da função de dados é proporcional ao número de DER e RLR que ela possui. A Tabela 3 apresenta esta relação.

Tabela 3. Complexidade das funções de dados

RLR \ DER	1 - 19	20 – 50	> 50
1	Baixa	Baixa	Média
2 – 5	Baixa	Média	Alta
> 5	Média	Alta	Alta

O número de pontos atribuídos a uma função de dados dependerá da complexidade da função e do seu tipo (AIE ou ALI), como apresentado na Tabela 4.

Tabela 4. Tamanho das funções de dados

Complexidade \ Tipo	ALI	AIE
Baixa	7	5
Média	10	7
Alta	15	10

A contagem das funções de transação segue o mesmo procedimento das funções de dados, definindo os tipos e a calculando a complexidade de cada transação. As funções transacionais são classificadas como CE (consulta externa), EE (entrada externa) ou SE (saída externa). Para definição do tipo de uma função transacional é preciso observar: (1) se é um processo que apenas consulta uma ou mais funções de dados sem realizar nenhum processamento; (2) se é um processo de consulta com algum processamento; ou (3) se é um processo que processa dados para alimentar um ou mais ALI. No primeiro caso, a função transacional é denominada consulta externa (CE); no segundo, ela é uma saída externa (SE); e, no último caso, é uma entrada externa (EE).

Tabela 5. Complexidade das funções de transação

EE			
ALR \ DER	**1 - 4**	**5 - 15**	**> 15**
0 - 1	Baixa	Baixa	Média
2	Baixa	Média	Alta
> 2	Média	Alta	Alta

SE e CE			
ALR \ DER	**1 - 5**	**6 - 19**	**> 19**
0 - 1	Baixa	Baixa	Média
2 - 3	Baixa	Média	Alta
> 3	Média	Alta	Alta

O cálculo da complexidade de uma função de transação é feito a partir do número de arquivos lógicos referenciados (ALR) e de dados elementares referenciados (DER) pela mesma, conforme mostra a Tabela 5. Um ALR é definido como uma

"função de dados consultada e/ou mantida por uma função transacional". O DER segue a mesma definição usada nas funções de dados. Com a complexidade e o tipo de uma função de transação, é possível definir o seu número de pontos de função, como apresentado na Tabela 6.

Tabela 6. Tamanho das funções de transação

Complexidade \ Tipo	EE	SE	CE
Baixa	3	4	3
Média	4	5	4
Alta	6	7	6

Com as funções de dados e de transação medidas, faz-se o somatório dos pontos atribuídos as mesmas para obter o número de pontos de função do sistema. Este pode ser utilizado como medida de esforço para o desenvolvimento do projeto de software.
Como pode ser observado, o número de pontos de função (esforço) depende do número e complexidade dos requisitos de um software, sendo esta complexidade determinada por tabelas de referência baseadas em dados que descrevem suas transações. Estas tabelas são caracterizadas por intervalos discretos, de modo que um pequeno aumento na complexidade de uma transação não necessariamente aumenta a estimativa de esforço para o seu desenvolvimento. O uso destes intervalos discretos, assim como as dependências existentes entre as transações e funções de dados, cria não-linearidades na estimativa de esforço a medida que novos requisitos são adicionados ou extraídos de um conjunto inicial. Estas não linearidades dificultam a seleção do conjunto ideal de requisitos para uma *release* e criam oportunidades para o uso de algoritmos de busca heurística para resolver o NRP.

3.2 Aplicando a Análise de Pontos de Função ao Problema da Próxima *Release*

Com base na definição clássica do NRP (Seção 2.2), este trabalho introduz os conceitos da Análise de Pontos de Função ao NRP. Os diferenciais da nossa proposta em relação à formulação clássica do NRP são: (1) considerar as dependências entre os elementos da APF, conforme mostrado na Seção 3.1; (2) utilizar a APF como base para o cálculo do custo dos requisitos; e (3) considerar as funções de dados e transacionais de acordo com os interesses dos patrocinadores pelos requisitos (v_{ij}, na definição padrão).

Neste sentido, a seleção de requisitos começa com um conjunto de funções de dados e transação representando todo o escopo de software, ou seja, todos os requisitos desejados pelos patrocinadores. A Tabela 7 sumariza a notação dos elementos envolvidos na APF.

Tabela 7. Tabela de notações dos elementos de APF

	Tamanho	Definição	Representação
S	*k*	Conjunto de patrocinadores envolvidos no projeto	$\{s_1, s_2, ..., s_k\}$
T	*n*	Conjunto das funções de transação do software	$\{t_1, t_2, ..., t_n\}$
D	*m*	Conjunto das funções de dados do software	$\{d_1, d_2, ..., d_m\}$
tipo(d_i)	-	Tipo da função de dados d_i	ALI ou AIE
RLR(d_i)	p_i	Conjunto das RLR de uma função de dados d_i	$\{rl_{i1}, ..., rl_{ipi}\}$
DER(rl_i)	q_i	Conjunto dos DER de um RLR rl_i	$\{dr_{i1}, ..., dr_{iqi}\}$
tipo(t_i)	-	Tipo da função de transação t_i	EE ou SE ou CE
DER(t_i)	r_i	Conjunto de DER referenciados por t_i	$\{dr_{i1, ..., } dr_{ii}\} \therefore$ $dr_{ij} \in \bigcup_{d \in D} \bigcup_{rl \in RLR(d)} DER(rl)$
PR(t_i)	-	Conjunto de transações que precedem a transação t_i	$\{t_x, ..., t_y\}$
v_{ij}	-	Interesse de um patrocinador s_j na transação t_i	-
p_j	-	Importância de um patrocinador s_j na empresa	-

A partir das notações acima é possível representar o número de pontos de função de uma função de dados $d_i \in D$ com base no seu tipo, seus RLR e os DER mantidos por estes RLR. O conjunto dos DER da função de dados d_i é calculado pela equação abaixo.

$$DER(d_i) = \bigcup_{rl \in RLR(d_i)} DER(rl) \tag{11}$$

O número de pontos de função de d_i é representado por *PFD(tipo(d_i), RLR(d_i), DER(d_i))* e calculado conforme as tabelas apresentadas na Seção 3.1. A fórmula abaixo representa o número total de pontos de função de dados do software.

$$PFD = \sum_{d_i \in D} PFD(tipo(d_i), RLR(d_i), DER(d_i)) \tag{12}$$

De forma similar, é possível representar o número de pontos de função de uma função de transação $t_j \in T$ com base no seu tipo, as funções de dados referenciadas por ela e os DER usados por esta função de transação. O conjunto de funções de dados referenciadas por t_j, *ALR(t_j)*, é calculado pela equação abaixo.

$$ALR(t_j) = \{d_i \in D : \exists\, rl_i \in RLR(d_i) \rightarrow DER(rl_i) \cap DER(t_j) \neq \emptyset\} \tag{13}$$

Com isto, o número de pontos de função de t_j é representado por *PFT(tipo(t_j), ALR(t_j), DER(t_j))* e também é calculado conforme as tabelas apresentadas na Seção 3.1. O número de pontos de função das funções de transação do software é dado abaixo.

$$PFT = \sum_{t_j \in T} PFT\big(tipo(t_j), ALR\big(t_j\big), DER(t_j)\big) \tag{14}$$

Por fim, o número de pontos de função do software é dado pela fórmula abaixo.

$$PF = PFD + PFT \tag{15}$$

Conforme apresentado na seção 3.1, na modelagem da APF o elemento funcional para os patrocinadores é a função de transação, ou seja, é nela que o patrocinador tem interesse, sendo a função de dados decorrente da necessidade de armazenamento e/ou leitura de dados por uma ou mais funções de transação. Assim, para o particionamento do software em *releases* tomaremos um subconjunto de *T*, representado por *T'*, referente às funções de transação que serão consideradas na *release*.

Na modelagem da APF, as dependências podem ocorrer entre duas funções de transação e entre uma função de dados e uma função de transação. O primeiro tipo de dependência tem como efeito que, ao se escolher o subconjunto T' de funções de transação para uma *release*, é preciso identificar que outras funções de transação devem ser incluídas nesta *release* através de uma análise de precedência. Isto acontece porque uma função de transação pode ser funcionalmente dependente de uma ou mais funções de transação, ou seja, ela pode utilizar recursos providos por outras transações e ficar inviável sem que elas sejam previamente (ou conjuntamente) desenvolvidas. Por exemplo, a função de transação "Realizar Venda de Produto" é dependente da função de transação "Cadastrar Produto", pois não é possível vender produtos que não tenham sido previamente cadastrados no sistema. Este é o tipo mais comum de dependência entre requisitos, abordado em trabalhos anteriores (BAGNALL et al., 2001; ZHANG et al., 2013).

Quanto à dependência entre uma função de dados e uma função de transação, o manual de contagem de pontos de função (IFPUG, 2009) indica que é necessário contabilizar quais funções de dados (ALR) e quais de seus dados (DER) uma função de transação utiliza para determinar seu tamanho em pontos de função. Neste caso, a escolha das funções de dados para uma *release* depende das funções de transação escolhidas, ou seja, somente as funções de dados referenciadas por pelo menos uma

função de transação escolhida para a *release* serão consideradas. O mesmo ocorre com os DER de uma função de dados: somente os DER referenciados por alguma transação serão considerados como elementos da função de dados na *release*.

Por conta dos dois tipos de dependência, ao selecionar o subconjunto T' é preciso: (1) adicionar as funções de transação que precedem, diretamente e indiretamente, as transações selecionadas; (2) encontrar os DER referenciados pelas funções de transação selecionadas e adicionadas; (3) encontrar as funções de dados e os RLR nas quais estes DER estão contidos; (4) contar o número de pontos de função das funções de dados considerando apenas os DER e RLR utilizados pelas funções de transação selecionadas; e (5) calcular o número de pontos de função final para a *release*.

Após a etapa (5), teremos o número total de pontos de função da *release*, que multiplicado pelo custo unitário do ponto de função gerará o custo total da *release*. Este custo, por sua vez, substituirá a equação (1) da formulação clássica do NRP (ver seção 2.1). É importante perceber que a otimização deste custo não é simples – uma transação retirada do escopo da *release* pode tornar diversos DER desnecessários, fazendo com que a complexidade da sua função de dados e, consequentemente, o custo da *release* sejam reduzidos. Assim, neste trabalho consideramos as não-linearidades do método de pontos de função, calculando o custo dos requisitos como uma caixa-branca.

Considerando o conjunto de funções de transação selecionadas *T'*, o conjunto de funções de transação que precedem diretamente *T'* é calculado pela equação abaixo.

$$precDiretos(T') = \bigcup_{t \in T'} PR(t) \qquad (16)$$

O conjunto de todas as funções de transação precedentes do conjunto T' é definido recursivamente pela fórmula abaixo.

$$precedentes(T') = precDiretos(T') \cup precedentes\big(precDiretos(T')\big) \qquad (17)$$

Finalmente, *T''*, que representa o conjunto de todas as funções de transação que devem ser consideradas na *release* quando escolhido o subconjunto T^i, é:

$$T'' = T' \cup precedentes(T') \qquad (18)$$

Para melhor entendimento da formulação recursiva, a Figura 4 ilustra três subconjuntos: cada elipse representa uma transação, sendo que as elipses tracejadas não estão selecionadas para a *release* em questão. Sabendo que as setas (→) representam a precedência das transações, temos que o lado esquerdo representa o conjunto T', o meio

representa *precedentes*(T') e o lado direito representa T". Sendo assim, $T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$, $PR(t_1) = \{t_6\}$, $PR(t_2) = \{t_1, t_4\}$, $PR(t_3) = \{t_2\}$, $PR(t_4) = \{t_5\}$, $PR(t_5) = PR(t_6) = \emptyset$, $T' = \{t_2, t_4\}$, $precedentes(T') = \{t_1, t_4, t_5, t_6\}$ e $T'' = \{t_1, t_2, t_4, t_5, t_6\}$.

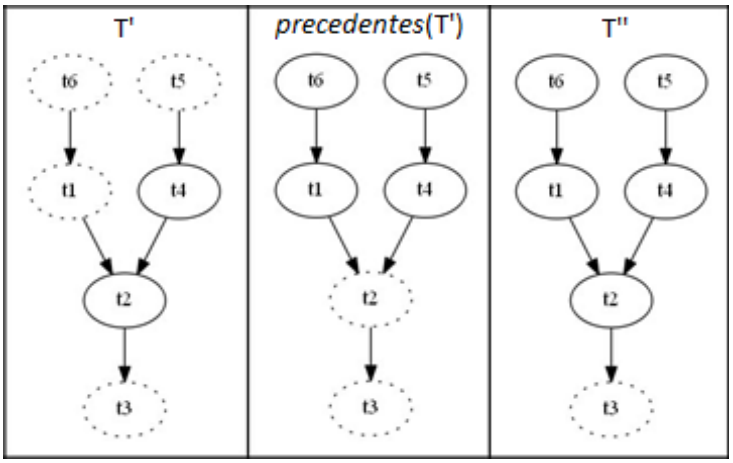

Figura 4. Conjuntos T', *precedentes*(T') e T"

Selecionado o subconjunto de transações *T"*, é preciso encontrar os DER utilizados por este subconjunto (*DR*), conforme a equação abaixo.

$$DR(T'') = \bigcup_{t_i \in T''} DER(t_i) \tag{19}$$

Em seguida, é preciso selecionar os RLR que mantém os DER referenciados pelas funções de transação. Estes RLR são representados pela equação RL(T'') abaixo. Estes RLR vão manter um subconjunto dos seus DER originais (*DR*), uma vez que DER não utilizados pelas funções de transação devem ser desconsiderados.

$$RL(T'') = \{rl_i \in \bigcup_{d_i \in D} RLR(d_i) \therefore DER(rl_i) \cap DR(T'') \neq \emptyset\} \tag{20}$$

$$DR(rl_i, T'') = DER(rl_i) \cap DR(T'') \tag{21}$$

Finalmente, precisamos selecionar as funções de dados que mantém os RLR que compõem o conjunto *RL*(T"). Isto é feito pela equação *D*(T'') abaixo. Estas funções de dados vão manter um subconjunto dos seus RLR originais, uma vez que RLR que não mantenham dados utilizados pelas funções de transação devem ser descartados.

$$D(T'') = \{d_i \in D : RLR(d_i) \cap RL(T'') \neq \emptyset\} \tag{22}$$

$$RLR(d_i, T'') = RLR(d_i) \cap RL(T'') \tag{23}$$

$$DER(d_i, T'') = DER(d_i) \cap DR(T'') \tag{24}$$

Com *T"*, *D(T")*, *RL(T")* e *DR(T")* é possível obter o total de pontos de função para a *release*. Para isto, é preciso calcular PFD' e PFT' referentes, respectivamente, ao

número de pontos das funções dados e das funções de transação consideradas na *release*. Abaixo, apresentamos as representações destes cálculos.

$$PFD' = \sum_{d_i \in D(T")} PFD(tipo(d_i), RLR(d_i, T"), DER(d_i, T")) \quad (25)$$

$$PFT' = \sum_{t_j \in T"} PFT\big(tipo(t_j), ALR\big(t_j\big), DER(t_j)\big) \quad (26)$$

A Figura 5 representa a utilização de uma função de dados d_1 (neste caso, um ALI) pelas funções de transação t_2 e t_3 e considera T = {t_2, t_3} e T"= {t_2}. Percebe-se que a transação t_2 utiliza os DER dr_1 a dr_7, dr_{16} e dr_{17}, enquanto a transação t_3 utiliza os DER dr_1, dr_2, dr_8 a dr_{23}. Considerando as duas transações, temos *PFD*(*tipo*(d_1), RLR(d_1), DER(d_1)) = 15, pois |RLR(d_1)| = 4 e |DER(d_1)| = 23. No entanto, ao eliminar t_3 do conjunto de transações selecionadas não precisaremos mais dos RLR rl_2 e rl_4, assim como de alguns DER de rl_3. Com isto, ficamos com *PFD*(*tipo*(d_1), RLR(d_1, *T"*), DER(d_1, *T"*)) = 10, pois |RLR(d_1, *T"*)| = 2 e |DER(d_1, *T"*)| = 9. Este exemplo demonstra que a retirada de uma função de transação pode provocar uma redução não-linear no esforço necessário para o desenvolvimento da *release*.

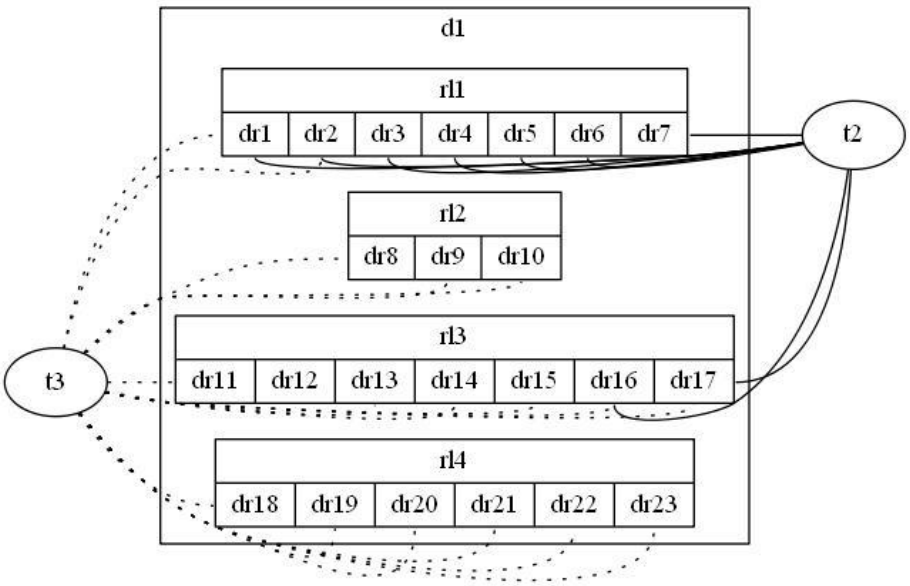

Figura 5. Utilização de d_1

A partir de PFD' e PFT' é possível calcular o esforço total, em pontos de função, para o desenvolvimento da *release*. Neste sentido, a formulação bi-objetivo da NRP sob a perspectiva da APF é representada pelas funções a seguir:

$$\text{Minimizar } f(1) = PFD' + PDT' \quad (27)$$

$$\text{Maximizar } f(2) = \sum_{j=1}^{k} p_j \sum_{i \in T''} v_{ij} \quad (28)$$

Como apresentado na Seção 2.1, na formulação do NRP mono-objetivo o esforço tem um limite superior fixado. Portanto, considerando *B* o número máximo de pontos de função que pode ser desenvolvido por *release*, é possível representar o NRP mono-objetivo sob a perspectiva da APF pelas seguintes funções.

$$PFD' + PDT' < B \quad (29)$$

$$\text{Maximizar } f(2) = \sum_{j=1}^{k} p_j \sum_{i \in T''} v_{ij} \quad (30)$$

3.3 Proposta de Solução

A seleção do subconjunto ideal de requisitos a ser implementado na próxima versão de um software é um problema NP-completo, mesmo quando não existem interdependências entre os requisitos (BAGNALL et al., 2001). No contexto do NRP sob a perspectiva da APF, o número de transações, funções de dados, interdependências e patrocinadores para uma instância do problema pode tornar o desenvolvimento de uma solução exata inviável, visto que o tempo para encontrar a melhor solução pode demorar de meses a anos.

Neste trabalho, propomos a utilização de buscas heurísticas, como as apresentadas no Capítulo 2, para reduzir o tempo necessário para encontrar uma boa solução, no caso de uma abordagem mono-objetivo para o problema, ou de um conjunto de boas soluções, no caso da abordagem bi-objetivo.

Para a abordagem mono-objetivo, propõe-se a utilização de heurísticas para gerar diversos conjuntos de funções de transação, sendo cada conjunto representado por *T',* conforme Seção 3.2. Para cada conjunto *T'* gerado, independente da heurística utilizada, um algoritmo compõe o conjunto *T"*, os DER, RLR e funções de dados de acordo com o critério de escolha apresentado na Seção 3.2. Após a escolha destes elementos, é calculado o número de pontos de funções referente a eles (função 1 da Seção 3.2), assim como a satisfação dos patrocinadores para o conjunto *T"* (função 2 da Seção 3.2). Estes dois valores servem de parâmetros para a busca heurística avaliar os resultados, de acordo com a formulação do problema (Seção 3.2).

A avaliação consiste em verificar se o número de pontos de função da solução é menor que o limite de investimento máximo desejado e comparar o valor da satisfação dos patrocinadores com alguma solução gerada anteriormente. Após esta comparação, a solução é classificada de acordo com critérios de cada heurística. A geração dos

conjuntos *T'* pela heurística é encerrada quando um critério de parada é atingido. Um exemplo de critério de parada é um número máximo de avaliações que a heurística deve realizar e este critério deve ser indicado quando se executa o processo de busca da solução. No final do processo de busca é apresentado um resultado, que na maioria das heurísticas é o melhor resultado encontrado durante as iterações da busca.

O processo de busca de boas soluções no contexto de uma abordagem bi-objetivo é similar à realizada pelo processo mono-objetivo. As diferenças estão na avaliação da solução e apresentação dos resultados. A avaliação da solução no processo de busca bi-objetivo consiste em verificar as dominâncias das soluções geradas pela heurística, conforme a definição de dominância apresentada no Capítulo 2. Para cada solução gerada, é avaliado se ela é não-dominada quando comparada a soluções geradas anteriormente e, a partir de então, ela é classificada de acordo com a heurística, podendo ser adicionada ao conjunto de boas soluções (fronteira de Pareto). Ao final, quando a busca atingir o critério de parada, tem-se como resultado a fronteira de Pareto, ou seja, um conjunto de soluções que podem ser selecionadas pelo tomador de decisão.

3.4 Considerações Finais

Este capítulo apresentou o problema da próxima *release* sob a perspectiva da Análise de Pontos de Função. Como base para formalização do problema, foram apresentados os conceitos da Análise de Pontos de Função. A formalização do problema foi abordada de duas maneiras: mono-objetiva e bi-objetiva. A abordagem mono-objetiva consiste em encontrar conjunto de transações, bem como os elementos das funções de dados utilizados por estas transações, que maximize a satisfação dos patrocinadores e atenda a uma restrição de esforço para desenvolvimento do software. Já a abordagem bi-objetiva consiste em encontrar um conjunto de boas soluções que apresentem um balanço entre o esforço requerido para a sua implementação e a satisfação dos patrocinadores, a fim de apresentá-las aos patrocinadores para escolha da que melhor atende às suas expectativas.

Propomos a utilização de heurísticas como solução do problema da próxima *release* sob a perspectiva da Análise de Pontos de Função. No entanto, o objetivo deste trabalho não é implementar um algoritmo de busca heurística específico para escolha de *T'*, mas avaliar diferentes algoritmos que possam ser aplicados para este fim. Foi parte deste trabalho desenvolver o algoritmo para a seleção de *T''* a partir dos DER e RLR decorrentes da escolha de *T'*, bem como o cálculo do número de pontos de função

destes elementos. Portanto, para escolha de *T'* é proposta a utilização do *framework JMetal* (DURILLO et al., 2010) que implementa diversas buscas heurísticas comumente utilizadas em trabalhos anteriores (vide Capítulo 2).

Neste trabalho avaliaremos o desempenho de diferentes algoritmos no contexto do NRP sob a perspectiva da APF. As métricas de desempenho, bem como o estudo experimental conduzido para esta avaliação, serão apresentadas no Capítulo 4.

Capítulo 4 - Avaliação do NRP baseado em Pontos de Função

Nesta seção, são descritos dois estudos experimentais. O primeiro tem como objetivo avaliar a proposta de solução ao Problema da Próxima Release sob a Perspectiva da Análise de Pontos de Função (NRP-APF), apresentada no capítulo anterior, e é conduzido através da comparação dessa proposta com a proposta clássica (NRP-CLS). Esta comparação é feita usando um algoritmo genético mono-objetivo para ambas propostas. O outro estudo tem como propósito comparar três algoritmos, sendo eles NSGA-II (DEB et al., 2002), SPEA2 (ZITZLER et al., 2001) e Aleatório, quando executados no contexto do NRP-APF.

Neste sentido, este capítulo mostra dois processos experimentais, sendo um para avaliação do NRP-APF em relação ao estado da arte e outro para comparação dos algoritmos que podem ser usados para encontrar soluções para o NRP-APF. Os dois estudos são apresentados em um mesmo capítulo por conta da grande quantidade de características que eles compartilham, inclusive a utilização das mesmas instâncias como insumo para execução e coleta de dados.

Cada estudo experimental foi conduzido segundo as etapas definidas em Wohlin et al. (2000): definição, planejamento, execução, análise e empacotamento do experimento. Os artefatos resultantes do empacotamento do experimento, que agregam todas as informações produzidas ao longo de todas as etapas do experimento, não são parte integrante desta Dissertação devido ao seu tamanho e por não serem essenciais para a descrição do experimento. Entretanto, estes artefatos podem ser obtidos no endereço http://vitorpadilha.github.io/otimizacao-pontos-de-funcao.

Este capítulo é dividido em cinco seções. A primeira apresentará as instâncias que são compartilhadas pelos dois experimentos. Os dois experimentos serão mostrados na segunda e terceira seção e equivalem, respectivamente, ao experimento para avaliação do NRP-APF e ao experimento para comparação dos algoritmos no contexto da APF. Uma quarta seção discute a validade do experimento. E, por último, uma extensão do experimento anterior avalia o comportamento do NRP-APF, comparado ao NRP-CLS, quanto as complexidades das funções de dados de um software.

4.1 Instâncias

Para condução dos estudos experimentais foi necessário encontrar instâncias que serviram como dados de entrada para execução dos algoritmos. Visando reduzir as ameaças à validade dos experimentos, optou-se pela escolha de instâncias reais. Instâncias artificiais poderiam ter propriedades distintas das instâncias reais, reduzindo a capacidade de generalização deste estudo. Porém, encontrar instâncias reais que atendessem aos requisitos necessários para execução dos estudos planejados foi um desafio, pelas seguintes razões:

- Não existe um padrão (*layout*) para contagem de pontos de função. Cada empresa utiliza o seu próprio padrão;
- Grande parte das contagens de pontos de função não relacionam quais funções de dados são utilizadas pelas funções de transação e somente indicam quantas funções de dados são utilizadas. Consequentemente, encontrar instâncias nas quais os DER utilizados pelas funções de transação são identificáveis, como nossa proposta requer, é muito difícil;
- Não encontramos nenhuma empresa que documenta a relação e grau de interesse dos patrocinadores por cada função de transação;
- Não há costume de relacionar as dependências entre as funções de transação. Estas dependências geralmente são intuitivas e são definidas no planejamento do projeto de software.

Apesar dos desafios, foram identificadas quatro instâncias produzidas em três ambientes distintos (empresas e instituições), três das quais contaram com a participação dos envolvidos neste estudo. A Tabela 8 apresenta estas quatro instâncias.

Tabela 8. Características das instâncias usadas no experimento

Nome	Nº de Função de Transação	Nº de Função de Dados	Nº de Patrocinadores	Nº de PF
ACAD	39	7	3	185
PSOA	65	9	3	290
PARM	98	21	3	451
BOLS	200	40	6	1131

A primeira instância (**ACAD**) é referente a um sistema acadêmico para controle de cadastro de turmas, alunos e professores de uma instituição de ensino, bem como matrículas dos alunos desta instituição.

A segunda instância (**PSOA**) é de um projeto de software para gestão de pessoas. Este projeto foi desenvolvido e implantado, sendo utilizado atualmente para gestão de pessoas em uma grande empresa.

A terceira instância (**PARM**) é um sistema que centraliza informações de parâmetros para sistemas satélites, que precisam de informações compartilhadas por várias aplicações.

Para quarta instância (**BOLS**), um sistema de bolsa de valores para negociação de ativos ambientais, não foi possível obter todos os requisitos reais necessários para execução do experimento de acordo com a proposta apresentada neste trabalho. A documentação da contagem de pontos de função contemplava a relação entre os DER e funções de transação, porém não existia o relacionamento de dependência entre transações e os graus de interesse dos patrocinadores do projeto. Para efeito do estudo, consideramos que o sistema não possui nenhuma relação de dependência entre transações e para o grau de interesse dos patrocinadores foi feito uma geração aleatória relacionando seis patrocinadores às 200 funções de transação do sistema.

4.2 Estudo Experimental: Avaliação do NRP-APF

Nesta seção, apresentamos o primeiro estudo experimental para avaliação do NRP-APF. O objetivo deste estudo é comparar as soluções obtidas pela proposta mono-objetiva do NRP-APF, representada pelas equações 29 e 30 do capítulo 3, com as soluções obtidas pela solução do NRP-CLS (vide seção 2.2).

O resultado esperado deste estudo é que o NRP-APF produza melhores soluções quando comparado ao NRP-CLS do ponto de vista de satisfação dos patrocinadores dado um limite de investimento em uma nova *release* do software sob análise. Como discutido no capítulo anterior, na abordagem NRP-APF um requisito (neste caso, as funções de dados) pode ter seu esforço de implementação diminuído de acordo com outros requisitos (neste caso, as funções de transação) selecionados para a *release*.

4.2.1 Questões de Pesquisa

A avaliação da proposta ao NRP-APF foi conduzida a partir da comparação desta proposta com o NRP-CLS. Para esta comparação, foi utilizada a formulação mono-objetiva do problema para ambas as propostas. Neste sentido, o esforço (custo) disponível para implementação da *release* foi fixado e esperava-se que um algoritmo

heurístico de busca encontre uma solução que maximize a satisfação dos patrocinadores do projeto. Com isto, a seguinte questão de pesquisa foi o alicerce deste estudo.

Q1: Dado um limite de investimento em uma release de um projeto de software, o NRP-APF produz soluções que geram mais satisfação aos patrocinadores do projeto quando comparado ao NRP-CLS?

Como apresentado no capítulo 3, a equação 4 (satisfação dos patrocinadores na formulação do NRP-APF) é similar a equação 2 (satisfação dos patrocinadores na formulação do NRP-CLS), considerando que uma função de transação é um requisito do sistema. Apresentada a questão de pesquisa, é possível formular as hipóteses que serviram para a análise dos dados coletados. São elas:

Hipótese Nula ($H_{0.1}$): Não há diferença quanto à satisfação dos patrocinadores entre as soluções obtidas pelo NRP-CLS e o NRP-APF.

Hipótese Alternativa ($H_{1.1}$): As soluções obtidas pelo NRP-APF representam maior satisfação dos patrocinadores quando comparadas às soluções do NRP-CLS.

4.2.2 Projeto do Estudo Experimental

O experimento em questão pode ser caracterizado como *off-line* (WOHLIN et al., 2000), pois depende apenas da execução de algoritmos e não foi executado dentro de uma rotina real de trabalho. Além disso, as instâncias utilizadas no estudo, apresentadas na seção 4.1, foram elaboradas por profissionais e não foram produzidas exclusivamente para o experimento.

Para avaliação do NRP-APF, o algoritmo escolhido foi o Genético Generacional[1], já implementado pelo framework *JMetal*[2] (DURILLO et al., 2010). Este algoritmo foi utilizado para encontrar soluções para as formulações NRP-AFP e NRP-CLS para as instâncias selecionadas, sendo estas soluções posteriormente analisadas conforme descrito abaixo. O *JMetal* é um framework que disponibiliza diversos algoritmos de busca heurística, sendo necessário apenas modelar o problema de

[1] jmetal.metaheuristics.singleObjective.geneticAlgorithm.gGA
[2] http:// jmetal.sourceforge.net

otimização alvo do estudo. Para análise dos resultados foi utilizada a ferramenta estatística *R*[3] versão 2.15.1.

Sabendo que os algoritmos heurísticos de busca geram possíveis soluções e para cada uma dessas soluções é feita uma avaliação de seu valor ao problema (comumente chamada de função *fitness*), o mesmo número máximo de avaliações, representado por MA, foi utilizado para execução dos algoritmos nas duas formulações. Neste sentido, não está entre os objetivos deste estudo comparar o tempo de execução despendido pelos algoritmos para encontrar soluções segundo as diferentes formulações.

Os parâmetros utilizados para execução do algoritmo genético foram os mesmos utilizados por DURILLO et al. (2009), que compararam dois algoritmos genéticos distintos no contexto do NRP-CLS. Estes parâmetros foram: (1) tamanho da população inicial foi igual a quatro vezes o número de funções de transação, ou seja, (4 x |T|); (2) *single-point crossover* com probabilidade de 90%; (3) seleção por torneio; (4) mutação *bit-flip* com probabilidade de 1/N, onde N é o número de requisitos; e (5) MA (que representa o número máximo de avaliações) igual a duas vezes o número de funções de transação elevado ao quadrado, ou seja, $(4 \times |T|)^2$.

Para avaliar a questão de pesquisa, foram utilizados testes de inferência estatísticos. Para avaliar a possibilidade de utilização de testes paramétricos, os dados dos experimentos passaram pelo teste de normalidade *Kolmogorov-Smirnov*. Como nenhum dos conjuntos de dados tinha distribuição normal, utilizamos o teste estatístico não-paramétrico de *Wilcoxon-Mann-Whitney* para encontrar o valor de *p-value* (BARROS, 2012; SOUZA et al., 2011), cujo valor informa a probabilidade de dois conjuntos de dados apresentarem médias iguais. Outro indicador utilizado na comparação das duas formulações é o *effect-size* (BARROS, 2012) que dá o tamanho de efeito entre dois conjuntos de dados. Este tamanho de efeito permite identificar quantas vezes um conjunto é melhor que outro em uma comparação par-a-par a partir dos resultados coletados do experimento. O tamanho de efeito utilizado neste estudo foi calculado através da métrica não-paramétrica Â12 de Vargha e Delaney (VARGHA; DELANEY, 2000).

Neste estudo, o esforço (custo) disponível para a implementação da próxima *release* de um software foi fixado em nove faixas proporcionais (10%, 20%, ..., 90%) ao esforço máximo para desenvolvimento do software. Para cada esforço disponível

[3] http://www.R-project.org

fixado, o algoritmo genético foi executado 50 vezes, para cada proposta (NRP-APF e NRP-CLS) e instância, totalizando 3600 execuções. Considerando que cada cenário de avaliação é formado pelo conjunto de uma instância e um esforço disponível, neste estudo tivemos 36 cenários. O problema foi modelado como um problema de minimização no *JMetal*, de maneira que o objetivo fosse a negação da satisfação dos patrocinadores (satisfação dos patrocinadores x -1) quando o esforço de desenvolvimento de uma solução a ser avaliada fosse menor ou igual ao esforço disponível. Porém, nos casos em que a solução não atendesse a esta restrição, o objetivo foi calculado como a diferença entre o esforço para desenvolvimento da solução e o esforço disponível. Esta variação no cálculo da função objetivo serve para que o algoritmo genético convirja para soluções que atendam à restrição, pois quanto menor esta diferença, mais próximo ele estará de soluções que podem ser aceitas.

Neste experimento foi preciso adequar o NRP-CLS para que pudesse utilizar as mesmas instâncias do NRP-APF. A adequação foi feita de maneira a considerar as funções de dados e funções de transação como requisitos do sistema. Porém, somente as funções de transação são requisitos de interesse dos patrocinadores, como mostrado pela equação 30 e na questão de pesquisa **Q1**. Portanto, consideramos a utilização de uma função de dados por uma função de transação como uma dependência, assim como as dependências entre requisitos de um sistema: caso uma função de transação utilize uma função de dados, esta última será adicionada à *release*. Neste caso, a função de dados será adicionada com todos seus elementos (DER e RLR) e, sendo assim, o valor da função de dados será invariável.

4.2.3 Execução e Análise dos Resultados

Nesta seção serão analisados os resultados do experimento, através da comparação do NRP-APF com o NRP-CLS. Primeiramente, analisamos as médias de satisfação dos patrocinadores encontradas pelas duas propostas nas nove faixas de esforço disponível, apresentadas na Tabela 9. Nesta tabela é possível encontrar as médias e o desvio padrão, representado por "média ± desvio padrão", das duas propostas que devem ser analisadas e comparadas por instância e esforço disponível (10%, 20%, ..., 90%).

Tabela 9. Médias e desvios padrão obtidos após a execução do NRP-CLS e do NRP-APF para as instâncias ACAD, PSOA, PARM e BOLS

	ACAD		PSOA		PARM		BOLS	
%CT[4]	NRP-CLS	NRP-APF	NRP-CLS	NRP-APF	NRP-CLS	NRP-APF	NRP-CLS	NRP-APF
10%	10,4 ± 0,0	10,4 ± 0,3	14,9 ± 1,9	13,7 ± 2,3	15,0 ± 0,6	15,1 ± 0,5	19,9 ± 0,5	21,6 ± 0,4
20%	28,2 ± 1,0	28,0 ± 1,0	23,6 ± 0,7	26,5 ± 1,6	28,5 ± 0,7	28,6 ± 0,7	38,2 ± 0,6	38,6 ± 0,4
30%	38,7 ± 1,0	38,6 ± 0,9	36,8 ± 0,8	39,5 ± 0,9	43,3 ± 0,6	43,4 ± 0,6	51,2 ± 0,3	51,2 ± 0,3
40%	46,0 ± 1,0	45,8 ± 1,1	50,3 ± 0,8	51,6 ± 0,5	54,8 ± 0,6	55,0 ± 0,6	60,5 ± 0,4	60,6 ± 0,3
50%	57,0 ± 0,6	57,0 ± 0,6	60,7 ± 0,6	62,5 ± 0,6	64,7 ± 0,4	64,6 ± 0,5	68,0 ± 0,3	68,1 ± 0,2
60%	67,7 ± 0,1	67,7 ± 0,4	67,9 ± 2,1	70,6 ± 1,7	73,7 ± 0,3	73,9 ± 0,4	74,9 ± 0,3	75,0 ± 0,2
70%	79,6 ± 0,1	79,5 ± 0,2	78,5 ± 0,2	78,5 ± 0,0	82,8 ± 0,4	82,6 ± 0,7	81,7 ± 0,2	81,9 ± 0,2
80%	86,4 ± 0,5	86,5 ± 0,4	86,8 ± 0,0	86,8 ± 0,0	89,9 ± 0,1	89,9 ± 0,1	88,5 ± 0,2	88,6 ± 0,2
90%	93,3 ± 0,2	93,4 ± 0,1	95,1 ± 0,1	95,1 ± 0,1	96,4 ± 0,0	96,3 ± 0,1	94,8 ± 0,2	94,9 ± 0,2

Analisando os dados da Tabela 9, é possível observar que as médias da satisfação dos patrocinadores encontradas pelos métodos NRP-APF e NRP-CLS são bem próximas na maioria dos cenários. Por exemplo, para as instâncias ACAD e PARM as médias são muito próximas ou iguais em todos os cenários. Para as instâncias PSOA e BOLS, já é possível observar que as médias de satisfação dos patrocinadores encontradas pelo NRP-APF são maiores do que as encontradas pelo NRP-CLS, principalmente nas faixas em que há menor esforço disponível. Estas constatações podem ser observadas visualmente nos gráficos apresentados na Figura 6.

[4] Percentual sobre custo total da instância. O custo total (C.T.), em pontos de função, de cada instância é apresentado na Tabela 9.

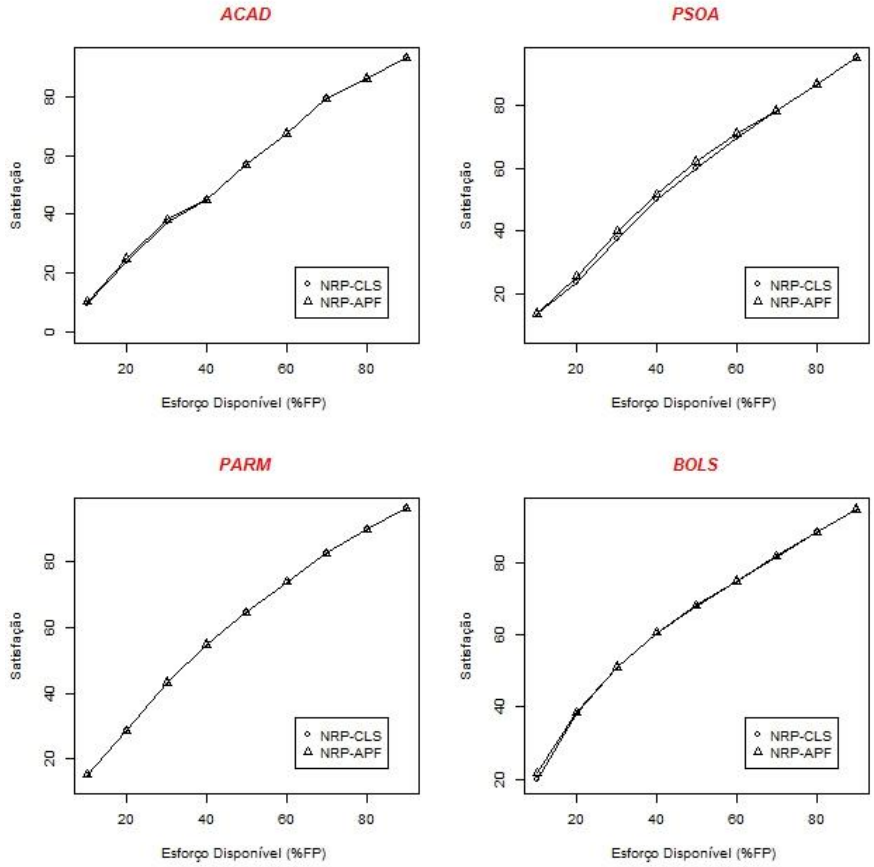

Figura 6. Satisfação dos patrocinadores por formulação do NRP para as instâncias ACAD, PSOA, PARM e BOLS

Como base para conclusões e validação dos resultados, as hipóteses apresentadas na seção 4.2.1 foram testadas utilizando os procedimentos estatísticos selecionados. Os valores de *p-values* e *effect-size* resultantes desta análise são mostrados na Tabela 10.

Tabela 10. *p-values* e *effects-sizes* para análise das propostas

	ACAD		PSOA		PARM		BOLS	
%CT	p-value	Effect-size (APF>CLS)	p-value	Effect-size (APF>CLS)	p-value	Effect-size (APF>CLS)	p-value	Effect-size (APF>CLS)
10%	0,253	55,4%	0,94	49,4%	0,904	50,9%	**<0,01**	98,9%
20%	0,496	55,1%	**<0,01**	75,2%	0,578	54,2%	**0,019**	67,6%
30%	0,371	56,3%	**<0,01**	100,0%	0,970	50,3%	0,201	40,4%
40%	0,927	50,7%	**<0,01**	94,8%	0,557	54,4%	0,178	60,1%
50%	0,438	45,0%	**<0,01**	95,2%	0,781	47,9%	0,095	62,3%
60%	0,986	49,9%	**<0,01**	93,2%	0,182	59,7%	0,094	62,4%
70%	0,582	46,9%	0,843	51,2%	0,483	44,8%	**<0,01**	74,3%
80%	0,925	49,3%	N\A	50,0%	0,018	34,4%	0,11	61,4%
90%	0,597	51,6%	N\A	50,0%	0,161	46,7%	**0,021**	66,4%

Analisando os dados apresentados na Tabela 10, onde os valores do *p-value* são quase sempre maiores que 0,05 e *effect-size* próximos a 50%, é possível confirmar que para quase todos os cenários a satisfação dos patrocinadores gerada nas duas formulações são similares. Os poucos cenários em que é possível afirmar com pelo menos 95% de certeza (ou seja, *p-value* < 0,05) que o NRP-APF produz satisfação dos patrocinadores maior do que o NRP-CLS são na instância PARM com esforço disponível entre 20% e 60% do esforço total, e na instância BOLS, com esforço disponível de 10%, 20%, 70% e 90% do esforço total. Portanto, somente é possível refutar a hipótese nula $\mathbf{H_{0,1}}$ e aceitar $\mathbf{H_{1,1}}$ nestes cenários.

Para explicar este resultado, devemos retornar ao modelo do problema. Como mostrado no Capítulo 3, uma função de dados pode sofrer variação no esforço esperado para o seu desenvolvimento em pontos de função de acordo com as funções de transação selecionadas. Esta variação pode diminuir a complexidade da função de dados ou transformar uma função de dados ALI em AIE. Com estas duas premissas, analisamos as instâncias ACAD e PARM e verificamos as seguintes características que não contribuíram para uma vantagem significativa do NRP-APF sobre o NRP-CLS para estas duas instâncias:

- Na instância PARM, todas as funções de dados são de baixa complexidade e, sendo assim, não foi possível diminuir a complexidade de nenhuma função de dados em decorrência da seleção de determinadas funções de transação, redução esta que permitiria uma vantagem do NRP-APF sobre o NRP-CLS;
- O conjunto de funções de dados da instância ACAD é composto por seis funções de complexidade baixa e uma de complexidade média. Todas as transações que utilizavam a função de dados de complexidade média dependiam da transação de inclusão desta função de dados, que já utilizava todos os DER da mesma, não permitindo assim diminuir sua complexidade.

Assim, a capacidade do NRP-APF mono-objetivo produzir melhores soluções quando comparado ao NRP-CLS quanto à satisfação dos patrocinadores depende de características da instância. O NRP-APF apresenta vantagens em sistemas que possuem funções de dados com complexidade média ou alta. Em sistemas em que grande a maioria das funções de dados é de baixa complexidade, o NRP-APF tende a produzir resultados similares ao NRP-CLS. Porém, em nenhum dos cenários se observou que o NRP-CLS foi melhor que o NRP-APF. Com isto, não é preciso analisar as

características das instâncias para se escolher o NRP-APF como método para escolha de requisitos de uma *release*, pois no pior caso ele será igual ao NRP-CLS.

Independente da instância, é possível discutir algumas vantagens do NRP-APF. Por exemplo, considere uma *start-up* de software com pouca capacidade de investimento e que precisa de capital para desenvolvimento. Esta empresa poderia usar o NRP-APF para priorizar os requisitos a ser incluídos da primeira *release* do seu software, maximizando a satisfação dos seus usuários dentro do limite dos recursos a ser investidos no desenvolvimento. Um cenário similar ocorre para uma empresa contratante de serviços de software, cujo objetivo é diminuir o custo total do projeto. Neste caso, escolher o NRP-APF como método para selecionar os requisitos do projeto significa ter um nível de satisfação dos patrocinadores maior ou igual ao que poderia ser atingido pelo NRP-CLS. Já para uma empresa a ser contratada, utilizar o NRP-APF significa ter maior capacidade de negociação de contratos com clientes e maior competitividade em licitações do governo, onde a APF é bastante utilizada. Adotando o NRP-APF, ela seria capaz de entregar os requisitos mais importantes nas primeiras *releases*, de acordo com o pedido pelo cliente, e satisfazê-lo de maneira que em contratos futuros ele priorize os serviços da empresa.

4.2.4 A Relação entre a Complexidade das Funções de Dados e o NRP-APF

Como discutido anteriormente, a vantagem de escolha do NRP-APF em relação ao NRP-CLS dependerá de características da instância, principalmente a complexidade das suas funções de dados. Quanto maior o número de funções de dados de média e alta complexidade, maiores serão as vantagens do NRP-APF sobre o NRP-CLS.

Nesta seção, é apresentada uma extensão do estudo apresentado na seção anterior. Nesta variação, a instância ACAD foi selecionada e suas funções de dados foram agrupadas (RLR e DER) de maneira a produzir configurações da mesma instância com funções de dados de complexidade distinta e avaliar o efeito desta reorganização na comparação entre o NRP-APF e NRP-CLS. Escolhemos a instância ACAD por ela não ter apresentado divergência na satisfação dos patrocinadores na comparação inicial e também por ser uma instância menor e, com isto, mais fácil de manipular suas funções de dados. Foram criadas três novas instâncias (ACAD2, ACAD3 e ACAD4) a partir da instância original. Sabendo que as funções de transação e os patrocinadores são os mesmos, as quatro instâncias apresentam as configurações listadas na Tabela 11.

Tabela 11. Instâncias geradas a partir da instância ACAD

Nome	Nº de Funções de Dados de Baixa Complexidade	Nº de Funções de Dados de Média Complexidade	Nº de Funções de Dados de Alta Complexidade	Nº de PF
ACAD	6	1	0	185
ACAD2	2	1	1	169
ACAD3	2	0	2	169
ACAD4	3	2	0	171

A reorganização das funções de dados da instância ACAD alterou também a contagem das funções de transação de modo que o número de ALR utilizados por uma função de transação teve que ser atualizada após a reorganização. Por este motivo, as instâncias ACAD2 e ACAD3 possuem o mesmo número de pontos de função.

Com estas novas instâncias, utilizamos o mesmo projeto de estudo apresentado na Seção 4.2.2 para execução e análise dos dados. A Tabela 12 apresenta as médias e desvios padrão das soluções encontradas pelo algoritmo genético nas formulações NRP-APF e NRP-CLS. Esta tabela possui a mesma representação da Tabela 9.

Tabela 12. Médias e desvios padrão obtidos após a execução do NRP-CLS e do NRP-APF para instâncias geradas a partir da instância ACAD

	ACAD		ACAD2		ACAD3		ACAD4	
%CT	NRP-CLS	NRP-APF	NRP-CLS	NRP-APF	NRP-CLS	NRP-APF	NRP-CLS	NRP-APF
10%	10,4 ± 0,0	10,4 ± 0,3	5,1 ± 0,9	9,6 ± 2,2	4,2 ± 2,2	9,6 ± 1,3	7,9 ± 1,4	10,0 ± 1,5
20%	28,2 ± 1,0	28,0 ± 1,0	20,0 ± 4,2	20,9 ± 2,0	17,2 ± 1,4	21,6 ± 0,6	24,0 ± 2,7	24,1 ± 2,9
30%	38,7 ± 1,0	38,6 ± 0,9	26,9 ± 2,1	32,8 ± 2,0	29,1 ± 1,2	32,5 ± 1,8	32,9 ± 2,4	34,6 ± 2,7
40%	46,0 ± 1,0	45,8 ± 1,1	37,5 ± 2,0	43,2 ± 1,4	37,3 ± 1,4	47,4 ± 0,6	44,1 ± 2,6	46,5 ± 1,6
50%	57,0 ± 0,6	57,0 ± 0,6	52,3 ± 1,1	55,8 ± 1,0	52,7 ± 1,0	57,8 ± 1,0	59,0 ± 3,0	57,3 ± 2,9
60%	67,7 ± 0,1	67,7 ± 0,4	67,0 ± 0,4	67,3 ± 0,4	64,1 ± 0,4	69,9 ± 0,5	70,1 ± 0,2	69,9 ± 0,5
70%	79,6 ± 0,1	79,5 ± 0,2	79,5 ± 0,2	79,5 ± 0,3	78,2 ± 0,5	81,4 ± 0,3	82,4 ± 0,3	82,3 ± 0,3
80%	86,4 ± 0,5	86,5 ± 0,4	86,6 ± 0,4	86,6 ± 0,3	86,6 ± 0,2	87,3 ± 0,3	88,5 ± 0,3	88,5 ± 0,2
90%	93,3 ± 0,2	93,4 ± 0,1	92,4 ± 0,1	92,4 ± 0,2	92,4 ± 0,0	93,4 ± 0,0	94,3 ± 0,0	94,3 ± 0,0

A partir da Tabela 12, é possível verificar que na maioria dos cenários (instância x esforço máximo) o NRP-APF produziu soluções que satisfazem mais os patrocinadores do que o NRP-CLS. Esta vantagem fica evidente quando as médias são apresentadas em um gráfico, representado pela Figura 7, que apresenta as médias de satisfação das propostas para cada instância gerada a partir de ACAD.

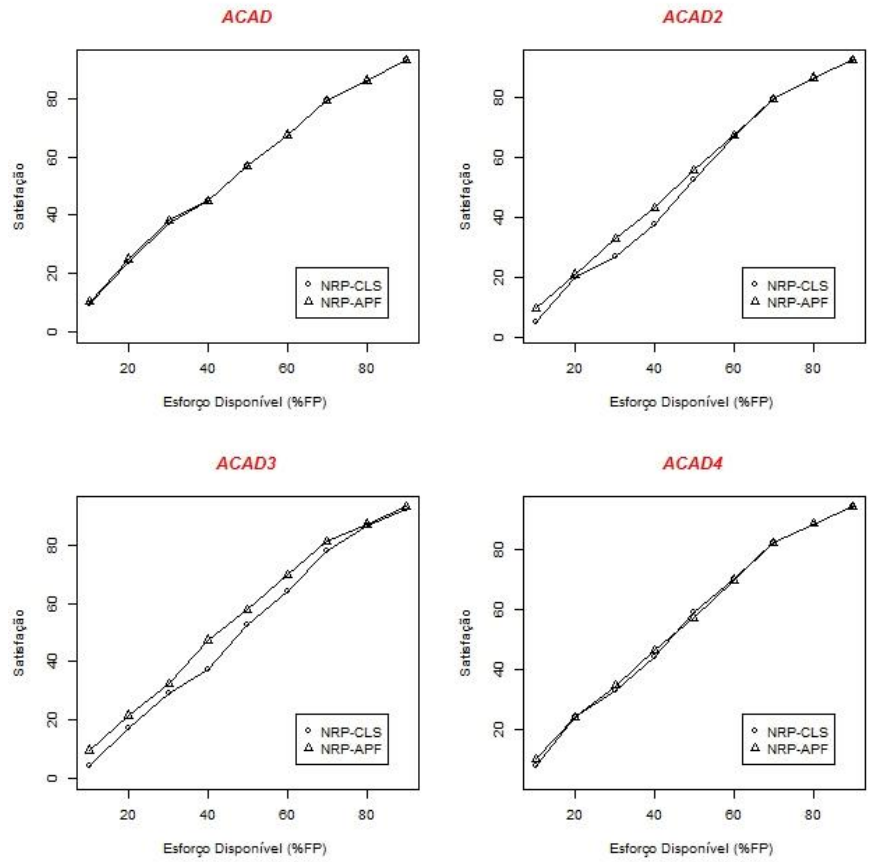

Figura 7. Satisfação dos patrocinadores por formulação para as instâncias ACAD, ACAD2, ACAD3 e ACAD4

A partir das configurações das instâncias, apresentadas na Tabela 11 e do gráfico representado pela Figura 7 é possível concluir que as instâncias que apresentaram maior diferença entre as médias observadas nas soluções encontradas pelo NRP-APF e o NRP-CLS são aquelas que apresentavam funções de dados mais agrupadas, ou seja, que continham mais DER e RLR e, consequentemente, maior complexidade. Esta conclusão podem ser confirmada a partir da análise de *p-values* e *effect-size* das satisfações obtidas pelas duas formulações para as instâncias alteradas de ACAD, como apresentado na Tabela 13.

Tabela 13. *p-values* e *effects-sizes* para análise da relação de complexidade das funções de dados com a vantagem do NRP-APF sob o NRP-CLS

	ACAD		ACAD2		ACAD3		ACAD4	
%CT	p-value	Effect-size (APF>CLS)	p-value	Effect-size (APF>CLS)	p-value	Effect-size (APF>CLS)	p-value	Effect-size (APF>CLS)
10%	0,253	55,40%	**<0,01**	90,70%	**<0,01**	100,00%	**<0,01**	96,20%
20%	0,496	55,10%	0,589	53,60%	**<0,01**	100,00%	0,191	40,40%
30%	0,371	56,30%	**<0,01**	96,40%	**<0,01**	92,60%	**<0,01**	84,80%
40%	0,927	50,70%	**<0,01**	99,70%	**<0,01**	100,00%	0,550	45,70%
50%	0,438	45,00%	**<0,01**	99,90%	**<0,01**	100,00%	0,233	41,10%
60%	0,986	49,90%	**0,019**	66,40%	**<0,01**	100,00%	0,633	52,80%
70%	0,582	46,90%	0,477	46,20%	**<0,01**	100,00%	0,871	49,00%
80%	0,925	49,30%	0,801	51,30%	**<0,01**	95,70%	0,87	48,90%
90%	0,597	51,60%	1,000	49,90%	**<0,01**	100,00%	N\A	50,00%

A partir da Tabela 13, juntamente com a Tabela 12, é possível afirmar, com grau de confiança de 99% (*p-value* < 0,01), que o NRP-APF produziu melhores resultados que o NRP-CLS para todos os cenários da instância ACAD3. Isto aconteceu na instância que possui as funções de dados mais complexas (duas funções de dados de alta complexidade). Além disto, para todos os cenários em que *p-value* < 0,05, ou seja, assumindo assim um grau de confiança de 95%, o NRP-APF teve médias maiores que o NRP-CLS e Effect-size (APF>CLS) maiores que 50%. Com Effect-size (APF>CLS) maior que 50%, pode-se concluir que o NRP-APF produziu melhores soluções na maioria das execuções do algoritmo.

Portanto, o resultado deste estudo com objetivo de analisar a relação entre a complexidade de funções de dados e a eficácia das formulações NRP-APF e NRP-CLS mostrou que quanto maior for a complexidade das funções de dados do software, maior será a vantagem do NRP-APF sobre o NRP-CLS. Sabendo disto, uma empresa que deseja contratar um serviço de desenvolvimento de software pode optar por especificar os requisitos de sistema de maneira a agrupar ao máximo as funções de dados. Com isto, utilizando o NRP-APF esta empresa poderá ter benefícios tanto na contenção de custos (caso tenha intenção de excluir requisitos do software), bem como na distribuição das entregas em *releases* (melhor satisfazendo os patrocinadores nas primeiras *releases*).

4.3 Avaliação de Algoritmos no Contexto do NRP-APF Bi-objetivo

Nesta seção apresentaremos um estudo experimental cujo objetivo é comparar três algoritmos de otimização multi-objetivo (NSGA-II (DEB et al., 2002), SPEA2 (ZITZLER et al., 2001) e Aleatório) no que diz respeito à sua capacidade de encontrar boas soluções para a formulação bi-objetivo do NRP-APF. Esta capacidade será avaliada comparando a qualidade das fronteiras de Pareto (conjunto de soluções não-dominadas) geradas pelos algoritmos, assim como o tempo de processamento necessário para produzir estas fronteiras.

Cada algoritmo recebe um número máximo de avaliações, que será utilizado como critério de parada na busca. Este critério servirá como base para análise do comportamento dos algoritmos, quanto ao tempo e qualidade das fronteiras geradas, a medida que se aumenta o número máximo de avaliações.

A inclusão do algoritmo Aleatório neste estudo se deve ao fato deste algoritmo não ter uma sistemática na escolha das soluções. Sendo assim, pode ser considerado como um "testador de sanidade" (ZHANG et al., 2007), permitindo avaliar se a pesquisa está sendo conduzida corretamente, pois em nenhum caso a busca aleatória deveria produzir melhores soluções que os demais algoritmos. Caso isto aconteça, há três possibilidades: (1) a pesquisa e suas métricas estão sendo feitas de maneira equivocada; (2) houve um erro de implementação do algoritmo que produzir piores resultados que a busca aleatória; ou (3) o algoritmo não é adequado ao problema.

4.3.1 Definições e Questões de Pesquisa

A análise das fronteiras de Pareto geradas por cada algoritmo é uma tarefa complexa, pois é preciso avaliar todos os conjuntos de soluções geradas e escolher qual conjunto é melhor entre conjuntos com grande número de soluções pode demandar muito tempo. Além da complexidade numérica da análise, existe o fator subjetivo. Um conjunto pode apresentar melhores soluções para determinada região no espaço de busca, enquanto outro encontra melhores soluções em outra região. Como dizer qual é o melhor conjunto? No caso do NRP, escolher entre menor investimento ou maior investimento dependeria de um acordo consensual entre os patrocinadores do projeto. Uma análise visual dos gráficos contendo esta fronteira, bem como o conjunto de soluções, pode indicar algumas conclusões, mas em muitos casos não é possível concluir qual fronteira é a melhor.

DURILLO et al. (2009) discutem como deve ser feita a avaliação da qualidade de uma fronteira de Pareto e cita as duas principais questões que devem ser consideradas: (1) minimizar a distância entre a fronteira de Pareto gerada e a fronteira ótima (convergência); e (2) maximizar a uniformidade da distribuição das soluções encontradas ao longo da fronteira (diversidade). Os autores sugerem a utilização de indicadores de qualidade que representam a convergência e/ou diversidade e definem uma métrica para avaliação das soluções geradas pela execução de um algoritmo multi-objetivo.

Neste trabalho, são considerados dois indicadores utilizados por DURILLO et al. (2009) para comparação de fronteiras de Pareto no contexto do NRP bi-objetivo: *Spread* e *Hipervolume*. Além destes indicadores, consideramos também o *Error Ratio* (VELDHUIZEN, 1999) e o *Generational Distance* (VELDHUIZEN; LAMONT, 1998). Nos próximos parágrafos, apresentaremos estes indicadores.

O *Error Ratio* é um indicador de qualidade que mede o número de acertos da fronteira de Pareto encontrada (Q) quando comparada à fronteira ótima (P). Trata-se de um indicador de convergência. Tal indicador foi apresentado por VELDHUIZEN (1999) e é representado pela fórmula (31). Nesta, é possível observar que quanto maior o número de acertos (soluções que pertencem a Q e a P), maior será o numerador. No caso em que todas as soluções de Q estão em P, o valor será zero. Portanto, quanto menor o *Error Ratio*, melhor será a fronteira de Pareto encontrada.

$$ER = 1 - \left(\frac{|\{d_i |\, d_i \in P \; e \; d_i \in Q \}|}{|Q|} \right) \tag{31}$$

De acordo com VELDHUIZEN e LAMONT (1998), o indicador *Generational Distance* (GD) serve para medir o quanto a fronteira de Pareto (Q) encontrada por determinado algoritmo está perto da fronteira de Pareto ótima (P). Assim como *Error Ratio*, *Generational Distance* é um indicador de convergência. Este indicador de qualidade pode ser representado pela fórmula (32), onde p é o número de objetivos sob os quais as soluções são analisadas e d_i é a distância Euclidiana (no espaço formado pelas funções objetivo) entre a solução i ∈ Q e a solução mais próxima de P. É possível verificar que quanto menor o valor de GD, melhor será a solução, pois a fronteira de Pareto representada pela solução estará mais próxima da fronteira ótima.

$$GD = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{\frac{1}{p}}}{|Q|} \quad (32)$$

Segundo DURILLO et al. (2010) e DURILLO et al. (2009), o *spread* (Δ) é um indicador de qualidade que mede a extensão de propagação da fronteira de Pareto encontrada pelo algoritmo, sendo assim um indicador de diversidade das soluções na fronteira. Ele pode ser representado pela fórmula (33), onde d_i é a distância Euclidiana entre duas soluções consecutivas, $\bar{d}$ é a média destas distâncias, d_i e d_f são a distância Euclidiana entre as soluções extremas da fronteira de Pareto ótima. Esta métrica tem valor zero para uma solução ideal (fronteira encontrada por um algoritmo é igual a fronteira ótima) e varia entre zero e um. Quanto menor o valor de Δ, melhor é a solução quanto ao seu *spread*.

$$\Delta = \frac{d_i + d_f + \sum_{i=1}^{N-1}\left|d_i - \bar{d}\right|}{d_i + d_f + (N-1)\bar{d}} \quad (33)$$

Como último indicador de qualidade, o *Hipervolume* (DURILLO et al., 2009) é um indicador qualitativo de convergência e diversidade. Esta métrica calcula o volume coberto pelas soluções componentes da fronteira de Pareto encontrada por um algoritmo (Q) no espaço definido pelas funções objetivo, sendo aplicável apenas quando todos os objetivos devem ser minimizados. Para cada solução i ∈ Q, um hipercubo v_i com o número de dimensões igual ao número de objetivos do problema sob análise é construído com base em um ponto de referência W e usando a solução *i* como diagonal oposta, conforme representa a Figura 8.

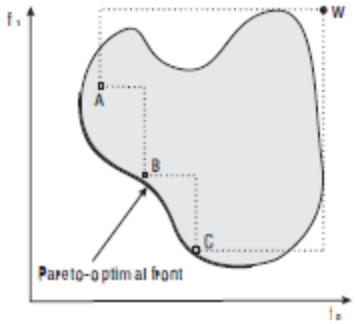

Figura 8. *Hipervolume* coberto pelas soluções não-dominadas (DURILLO et al., 2009)

O ponto de referência pode ser determinado construindo-se um vetor com os piores valores das funções objetivo. Posteriormente, é encontrada a união de todos os hipercubos e calculado seu *hipervolume*. Quanto mais próxima uma solução *i* estiver da fronteira de Pareto ótima e quanto melhor a distribuição de todas as soluções, maior o volume do hipercubo formado. O *hipervolume* é representado pela equação (34) e quanto maior o seu valor, melhor será a fronteira de Pareto encontrada.

$$HV = volume\left(\bigcup_{i=0}^{|Q|} v_i\right) \tag{34}$$

Como visto, os indicadores de qualidade dependem da fronteira de Pareto ótima para serem calculados. Porém, como é possível encontrar esta fronteira ótima para problemas cuja solução ideal é desconhecida? A alternativa mais segura é varrer todas as soluções possíveis no espaço de busca, encontrando com certeza a fronteira de Pareto ótima. No entanto, tal alternativa pode requerer um tempo inaceitável com os recursos computacionais atualmente disponíveis, tendo em vista o número de combinações possíveis e que deveriam ser avaliadas no espaço de busca. Portanto, é preciso achar alternativas viáveis que encontrem fronteiras próximas da ótima para análise das fronteiras geradas pelos diferentes algoritmos.

Neste trabalho, a fronteira de Pareto próxima da ótima utilizada como aproximação de P nas fórmulas dos indicadores de qualidade é obtida a partir das soluções não-dominadas (vide seção 2.4) quando são considerados os resultados das 150 execuções para cada instância (50 para cada algoritmo). Ou seja, as 150 fronteiras de Pareto geradas nestas execuções são comparadas e as soluções não-dominadas formam a fronteira de Pareto ótima da instância.

Após a apresentação dos indicadores de qualidade, podemos definir as seguintes questões de pesquisa.

Q2: Qual dos algoritmos produz fronteiras de Pareto com melhor *Error Ratio* no contexto do NRP-APF bi-objetivo?

Como descrito em trabalhos anteriores que tratam do NRP bi-objetivo (DURILLO et al., 2009, 2011; ZHANG et al., 2007), o algoritmo NSGA-II é o que apresenta menor divergência da fronteira ótima quando comparado a outros algoritmos. Portanto, espera-se neste trabalho comprovar que ele é melhor do que o SPEA2 e o

Aleatório quanto à convergência usando o indicador de qualidade *Error Ratio*. A análise será feita comparando o comportamento dos algoritmos em distintos números máximos de avaliações de função de *fitness* atribuídos aos algoritmos.

De forma análoga ao experimento anterior, duas hipóteses foram formuladas. Para os testes de hipóteses, não consideraremos o algoritmo aleatório, pois se espera que ele encontre soluções com qualidade muito inferior aos demais algoritmos. Porém, na análise dos indicadores, apresentaremos dados referentes às execuções dos três algoritmos, bem como análises estatísticas destes dados.

Hipótese Nula ($H_{0\text{-}ER}$): Não há diferença, quanto ao indicador *Error Ratio*, entre fronteiras de Pareto encontradas pelos algoritmos NSGA-II e SPEA2.

Hipótese Alternativa (**$H_{1\text{-}ER}$**): As fronteiras de Pareto encontradas pelo NSGA-II possuem menor *Error Ratio* que as fronteiras encontradas pelo SPEA2.

Q3: Qual dos algoritmos produz fronteiras de Pareto com melhor *Generational Distance* no contexto do NRP-APF bi-objetivo?

Assim como para o *Error Ratio*, espera-se comprovar que o NSGA-II é melhor do que o SPEA2 quanto à convergência, porém utilizando o indicador de qualidade de *Generational Distance*.

Hipótese Nula ($H_{0\text{-}GD}$): Não há diferença, quanto ao indicador *Generational Distance*, entre fronteiras de Pareto encontradas pelos algoritmos NSGA-II e SPEA2.

Hipótese Alternativa (**$H_{1\text{-}GD}$**): As fronteiras de Pareto encontradas pelo NSGA-II possuem menor *Generational Distance* que as fronteiras encontradas pelo SPEA2.

Q4: Qual dos algoritmos produz fronteiras de Pareto com melhor *Spread* no contexto do NRP-APF bi-objetivo?

Como descrito em trabalhos anteriores que tratam do NRP bi-objetivo (DURILLO et al., 2009, 2011; ZHANG et al., 2007), o algoritmo NSGA-II não apresenta uma boa diversidade em relação ao indicador de *Spread* quando comparado a outros algoritmos. Neste trabalho, espera-se observar tal comportamento e as hipóteses para o indicador *spread* são as seguintes:

Hipótese Nula ($H_{0\text{-}\Delta}$): Não há diferença, quanto ao indicador *Spread*, entre fronteiras de Pareto encontradas pelos algoritmos NSGA-II e SPEA2.

Hipótese Alternativa ($H_{1\text{-}\Delta}$): As fronteiras de Pareto encontradas pelo SPEA2 possuem menor *Spread* que as fronteiras encontradas pelo NSGA-II.

Q5: Qual dos algoritmos produz fronteiras de Pareto com melhor *Hipervolume* no contexto do NRP-APF bi-objetivo?

Em trabalhos anteriores (DURILLO et al., 2009, 2011; ZHANG et al., 2007), mostra-se que o NSGA-II apresenta um bom *Hipevolume* quando comparado a outros algoritmos. Considerando tal fato, neste estudo espera-se comprovar que o NSGA-II é melhor do que o SPEA2 quanto à convergência e diversidade utilizando o indicador de *Hipervolume* no contexto do NRP-APF. Portanto, as hipóteses são:

Hipótese Nula ($H_{0\text{-}HV}$): Não há diferença, quanto ao indicador *Hipervolume*, entre fronteiras de Pareto encontradas pelos algoritmos NSGA-II e SPEA2.

Hipótese Alternativa ($H_{1\text{-}HV}$): As fronteiras de Pareto encontradas pelo NSGA-II possuem maior *Hipervolume* que as fronteiras encontradas pelo SPEA2.

Q6: Qual dos algoritmos produz fronteiras de Pareto em menor tempo, considerando o mesmo número máximo de avaliações, no contexto do NRP-APF bi-objetivo?

O algoritmo Aleatório realiza pouco processamento para selecionar as soluções candidatas no espaço de busca. Por isto, espera-se que ele apresente melhor desempenho. Porém, como o objetivo é novamente comparar o NSGA-II e o SPEA2, o algoritmo Aleatório não constará na hipótese, servindo apenas como um teste de sanidade para ver o quanto mais lento ou rápido tais algoritmos estão comparado a uma referência. Portanto, os algoritmos serão executados utilizando o mesmo número máximo de avaliações e os tempos de busca serão avaliados. As hipóteses para esta questão de pesquisa são:

Hipótese Nula ($H_{0\text{-}TE}$): Não há diferença, quanto ao tempo de execução, para encontrar um conjunto de soluções entre os algoritmos NSGA-II e SPEA2 utilizando o mesmo número de avaliações.

Hipótese Alternativa ($\mathbf{H_{1\text{-}TE}}$): O tempo de execução do algoritmo NSGA-II é menor do que o SPEA2, quando utilizado o mesmo número de avaliações.

Hipótese Alternativa ($\mathbf{H_{2\text{-}TE}}$): O tempo de execução do algoritmo SPEA2 é menor do que o NSGA-II, quando utilizado o mesmo número de avaliações.

4.3.2 Projeto do Estudo Experimental

O experimento utiliza as mesmas instâncias e ferramentas para desenvolvimento e análise apresentada no primeiro estudo. Porém, os algoritmos utilizados são diferentes e também são implementados no *JMetal*. Neste estudo, foram utilizados os algoritmos NSGA-II[5], SPEA2[6] e Aleatório[7]. O *design* do experimento é similar, mas diferenciando que este experimento é caracterizado como um fator com três tratamentos, que são os algoritmos (NSGA-II, SPEA2 e Aleatório).

Os parâmetros dos algoritmos genéticos (NSGA-II e SPEA2) foram os mesmos utilizados no primeiro experimento. Para o SPEA2, é necessário um parâmetro extra que diz respeito ao tamanho do arquivo que armazena os resultados ao longo das gerações percorridas pelo algoritmo. Como no estudo de ZHANG et al. (2011), o tamanho do arquivo foi o mesmo da população inicial.

Também como no primeiro experimento, é preciso definir o número máximo de avaliações das funções objetivo que serão concedidas a cada algoritmo. Cada algoritmo foi executado 50 vezes e com número máximo de avaliações das funções objetivo igual a duas vezes o número de funções de transação elevado ao quadrado, ou seja, $(2 \times |T|)^2$. Porém, neste experimento, os algoritmos também foram executados com diferentes critérios de parada, sempre proporcionais ao número máximo de avaliações proposto. Estas proporcionalidades foram iguais a 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% e 90% de $(2 \times |T|)^2$. Para a quarta instância, os algoritmos também foram executados com proporcionalidades de 100% a 200% com intervalos de 10% para conclusões que serão mostradas na análise do experimento.

[5] jmetal.metaheuristics.nsgaII.NSGAII
[6] jmetal.metaheuristics.spea2.SPEA2
[7] jmetal.metaheuristics.randomSearch.RandomSearch

4.3.3 Execução e Análise do Experimento

O objetivo desta seção é apresentar a análise dos resultados obtidos a partir da execução dos três algoritmos multi-objetivo no contexto do NRP-APF. Para isto, dividiremos a seção em sete subseções. A primeira analisa graficamente a qualidade das fronteiras de Pareto geradas pela execução dos três algoritmos. As cinco subseções seguintes analisam, respectivamente, os resultados obtidos para os indicadores de qualidade *Error Ratio*, *Generational Distance*, *Spread* e *Hipervolume* e para o tempo de execução dos algoritmos. Por último, uma subseção voltada para as conclusões que podem ser extraídas dos resultados finaliza esta seção.

4.3.3.1 Análise das Fronteiras de Pareto Geradas Pelos Algoritmos

Nesta seção, são discutidas as fronteiras de Pareto geradas pelos algoritmos quando executados com número máximo de avaliações igual à $(2 \times |T|)^2$. Portanto, 600 fronteiras de Pareto foram geradas pelas execuções destes três algoritmos. Como o objetivo da avaliação das fronteiras de Pareto para esta seção é visual, a avaliação destas 600 fronteiras de Pareto é complexa. Neste sentido, foi escolhida a primeira execução de cada algoritmo para cada instância e foram gerados gráficos de dispersão que relacionam o esforço necessário para desenvolvimento da *release*, em número de pontos de função, e a satisfação dos patrocinadores, permitindo uma análise visual dos resultados. A Figura 9 apresenta estes gráficos. Com a análise visual dos gráficos nesta figura, é possível concluir que:

- O algoritmo SPEA2, geralmente, encontra as soluções que estão nas extremidades da fronteira e não são encontradas pelo NSGA-II;
- O algoritmo NSGA-II encontra um maior número de soluções, apresentando um conjunto de soluções mais denso para o mesmo problema;
- Não é possível concluir qual algoritmo é melhor apenas com base na análise visual, sem utilizar os indicadores de qualidade.

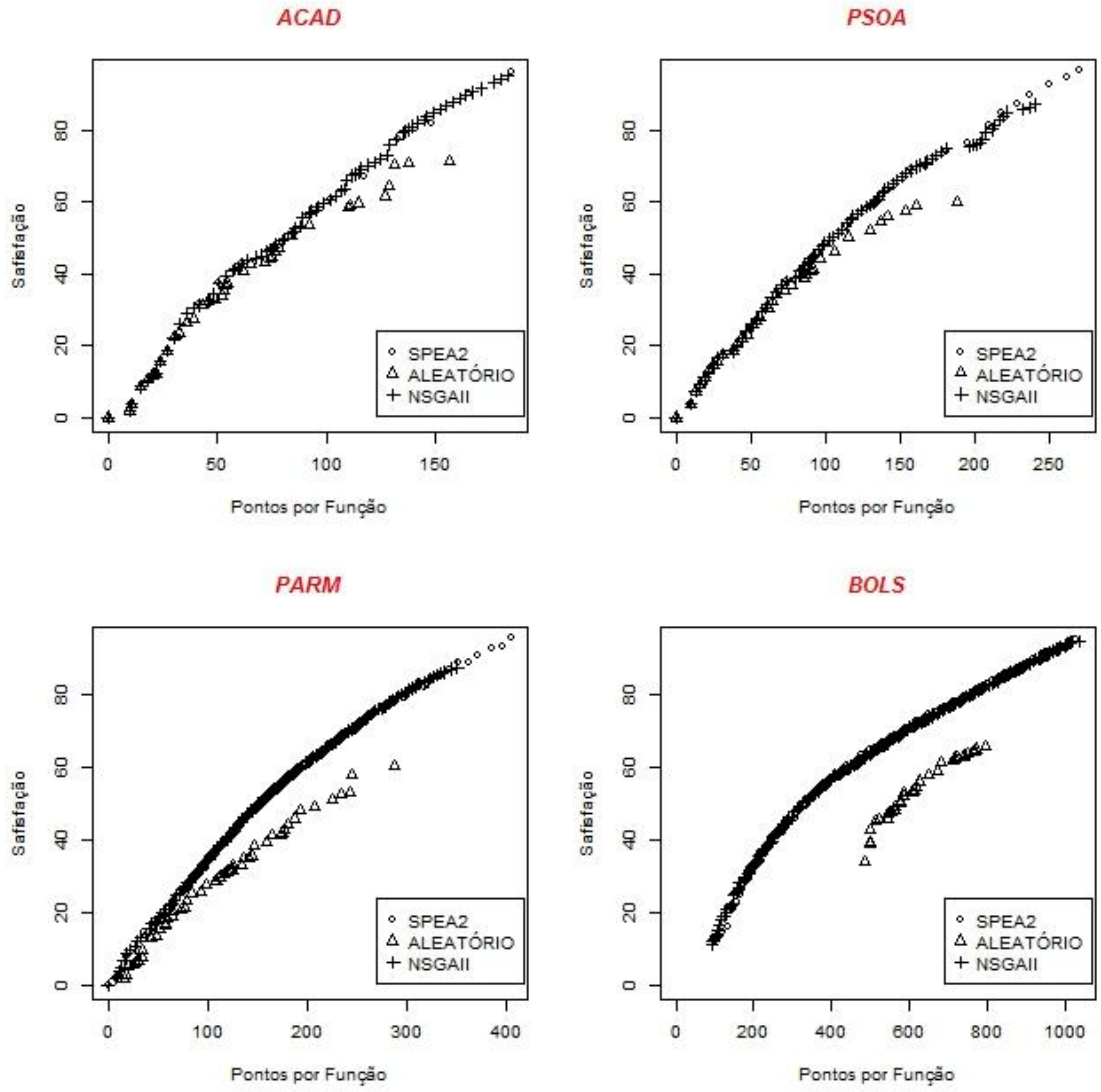

Figura 9. Gráfico de dispersão das execuções dos algoritmos sobre as quatro instâncias selecionadas

4.3.3.2 Análise do *Error Ratio*

A análise do *Error Ratio*, assim como para os outros indicadores de qualidade, é feita em três etapas. Na primeira etapa, é analisado um gráfico de linha que permite verificar o comportamento dos resultados quando relacionado ao número máximo de avaliações atribuídas aos algoritmos. Com isto, é possível ver este comportamento a medida que a restrição de número máximo de avaliações aumenta. Na segunda etapa, gráficos e tabelas apresentam dados referentes às execuções com número máximo de avaliações igual à $(2 \times |T|)^2$, que servirem para análises estatísticas e testes de hipóteses. Por último, apresentamos nossas conclusões para este indicador.

A Figura 10 apresenta as médias dos *Error Ratio* a medida que o número máximo de avaliações atribuído aos algoritmos aumenta. O número máximo de avaliações está representado como percentual de $(2 \times |T|)^2$.

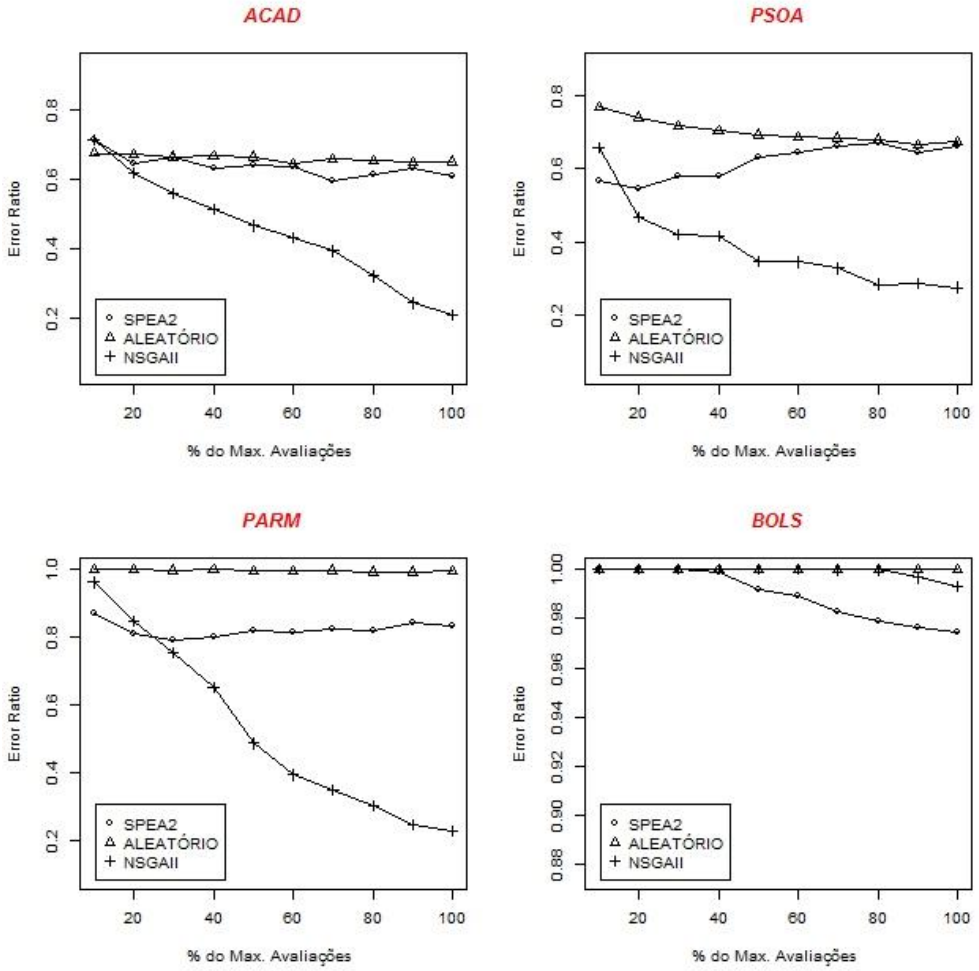

Figura 10. Evolução do *Error Ratio*

Sabendo que quanto menor o *Error Ratio*, melhor é a convergência da fronteira de Pareto, nas instâncias menores (ACAD, PSOA e PARM) é possível verificar que o SPEA2 produz melhores soluções (fronteira de Pareto) quanto ao *Error Ratio*, quando comparado ao NSGA-II e ao Aleatório, quando o número máximo de avaliações é baixo. Porém, é possível observar que o NSGA-II passa a produzir melhores resultados à medida que o número máximo de avaliações aumenta. Na quarta instância, isto não foi observado e o SPEA2 produziu melhores resultados até 160.000 avaliações. Isto ocorreu, provavelmente, pelo fato do número máximo de avaliações escolhido não ter sido suficientemente grande para que o NSGA-II passasse a produzir melhores soluções que o SPEA2.

Como ferramenta para reforçar as conclusões anteriores, a Figura 11 mostra um gráfico *boxplot* com dados de qualidade *Error Ratio* referentes ao número máximo de avaliações igual a $(2 \times |T|)^2$ e a Tabela 14 apresenta dados que servem para o teste das hipóteses elaboradas na Seção 4.3.1.

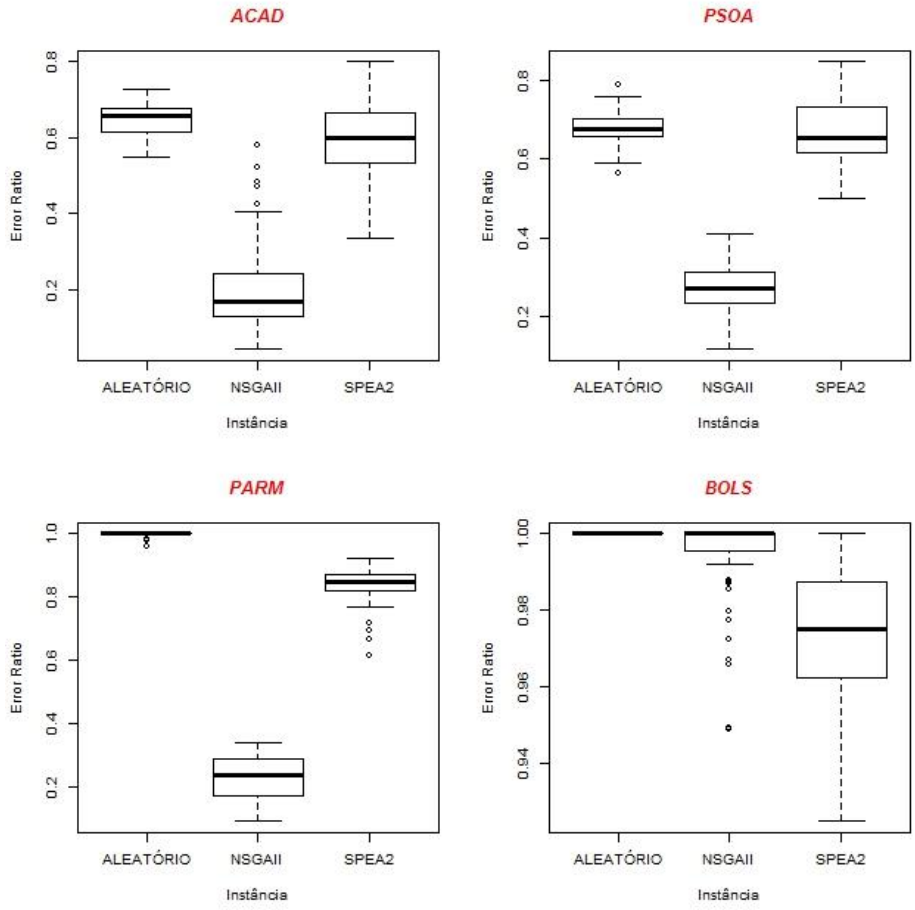

Figura 11. Gráficos *Boxplot* para *Error Ratio*

Como observado na Figura 10, os gráficos *boxplot* também confirmam que para as três instâncias menores o NSGA-II produziu melhores soluções quanto ao *Error Ratio*, quando o número máximo de avaliações é igual a $(2 \times |T|)^2$. Para a quarta instância, o SPEA2 é o algoritmo que produziu melhores resultados.

Tabela 14. Análise de indicador de qualidade: *Error Ratio*

Instância	Effect-Size			P-value		
-	SPEA2>NSGA-II	AL.>SPEA2	AL.>NSGA-II	SPEA2xNSGA-II	AL.xSPEA2	AL.xNSGA-II
ACAD	98%	62%	100%	<0,01	0,04	<0,01
PSOA	100%	58%	100%	<0,01	0,15	<0,01
PARM	100%	100%	100%	<0,01	<0,01	<0,01
BOLS	17%	95%	73%	<0,01	<0,01	<0,01

Pelos valores *p-value* apresentados na Tabela 14, pode-se afirmar com 99% de certeza que o NSGA-II produz resultados diferentes dos produzidos pelo SPEA2 e pelo Aleatório para todas as quatro instâncias quanto ao indicador *Error Ratio*. Neste caso, pode-se refutar $\mathbf{H_{0\text{-}ER}}$. No entanto, uma análise dos valores das médias dos gráficos *boxplot* da Figura 11 e pelo *effect-size* da mesma tabela, é possível aceitar $\mathbf{H_{1\text{-}ER}}$ somente para as três instâncias com menor número de funções de transação. Tal fato vai de encontro com a análise gráfica anteriormente descrita.

Portanto, a partir dos resultados apresentados, pode-se concluir apenas que o *Error Ratio* é melhor utilizando o NSGA-II ($\mathbf{H_{1\text{-}ER}}$) para as três instâncias menores deste experimento, não sendo possível generalizar para os demais contextos.

4.3.3.3 Análise do Generational Distance

A análise do indicador do *Generational Distance* tomará como base o mesmo tipo de análise feita para o *Error Ratio*. Neste sentido, a Figura 12 apresenta as médias dos *Generational Distance* à medida que o número máximo de avaliações atribuído aos algoritmos aumenta. O número máximo de avaliações está representado como percentual de $(2 \times |T|)^2$.

Sabendo que quanto menor o *Generational Distance*, melhor é a convergência da fronteira de Pareto, é possível verificar visualmente que, para as três instâncias maiores e com um menor número de avaliações, o algoritmo SPEA2 produziu fronteiras de Pareto com melhores *Generational Distance* quando comparado aos outros dois algoritmos. Também se pode observar para estas instâncias que a medida que o número máximo de avaliações aumenta, o NSGA-II tende a ser melhor que o SPEA2. Para a instância 4, não foi possível perceber tal vantagem do NSGA-II a partir deste gráfico.

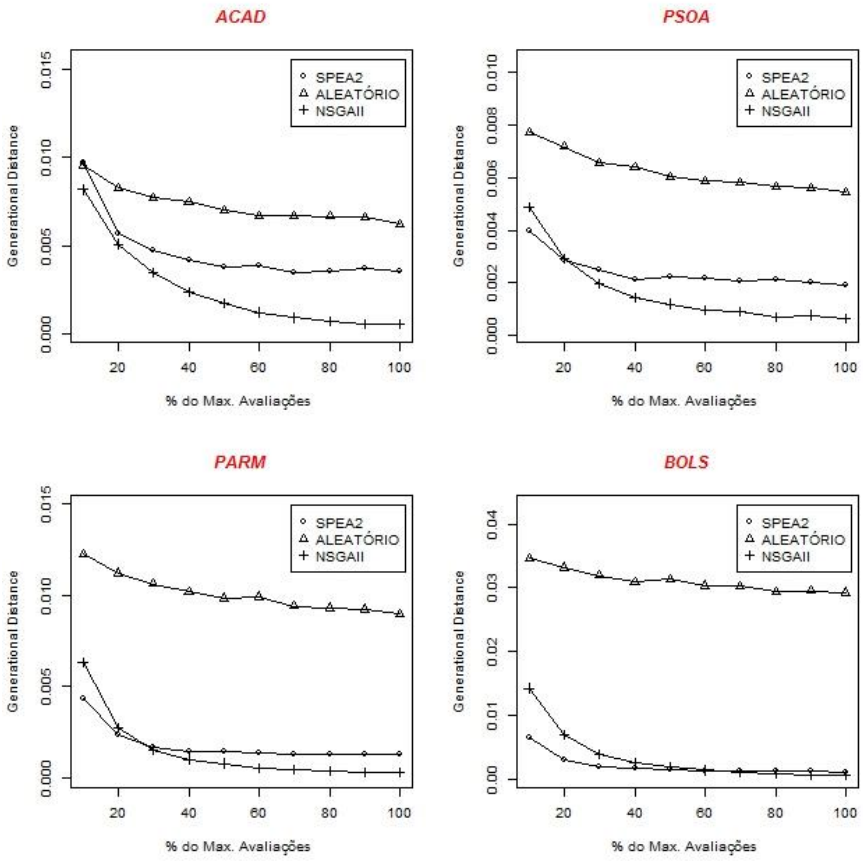

Figura 12. Evolução do *Generational Distance*

A Figura 13 mostra quatro gráficos *boxplot* com o indicador de qualidade *Generational Distance,* referentes ao número máximo de avaliações igual a $(2 \times |T|)^2$ e a Tabela 15 apresenta dados que servem para o teste das hipóteses elaboradas na Seção 4.3.1.

Como observado nos gráficos da Figura 12, os gráficos *boxplot*, representados na Figura 13, também confirmam que para as três instâncias menores o NSGA-II produziu melhores soluções quanto ao *Generational Distance*, quando o número máximo de avaliações é igual a $(2 \times |T|)^2$. Porém, nestes gráficos, é possível visualizar que para a quarta instância a vantagem do NSGA-II é muito pequena em relação ao SPEA2.

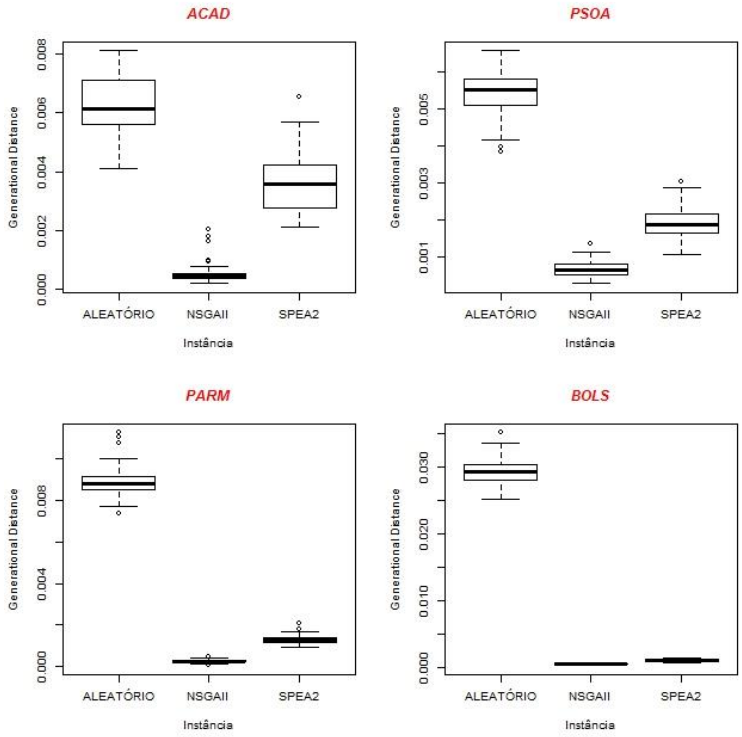

Figura 13. Gráficos *Boxplot* para *Generational Distance*

Tabela 15. Análise de indicador de qualidade: *Generational Distance*

Instância	Effect-Size			P-value		
	SPEA2>NSGA-II	AL.>SPEA2	AL.>NSGA-II	SPEA2xNSGA-II	AL.xSPEA2	AL.xNSGA-II
ACAD	100%	97%	100%	<0,01	<0,01	<0,01
PSOA	100%	100%	100%	<0,01	<0,01	<0,01
PARM	100%	100%	100%	<0,01	<0,01	<0,01
BOLS	100%	100%	100%	<0,01	<0,01	<0,01

A conclusão feita a partir da visualização dos gráficos pode ser confirmada pela análise dos valores de *p-values* da Tabela 15. A partir destes valores, é possível afirmar com 99% de certeza que as fronteiras de Pareto geradas pelo NSGA-II são diferentes, quanto ao *Generational Distance*, das geradas pelo SPEA2 e Aleatório, refutando assim a hipótese $\mathbf{H_{0\text{-}GD}}$ para todas as instâncias. Como apresentado na Figura 13, as médias das fronteiras geradas pelo NSGA-II foram menores e, com isto, é possível aceitar a $\mathbf{H_{1\text{-}GD}}$ para todas as instâncias. Portanto, diferentemente do *Error Ratio*, foi possível aceitar $\mathbf{H_{1\text{-}GD}}$ em todas as instâncias para o *Generational Distance*.

4.3.3.4 Análise do *Spread*

Como nas análises anteriores, a Figura 14 apresenta as médias dos *Spread* a medida que o número máximo de avaliações atribuído aos algoritmos aumenta.

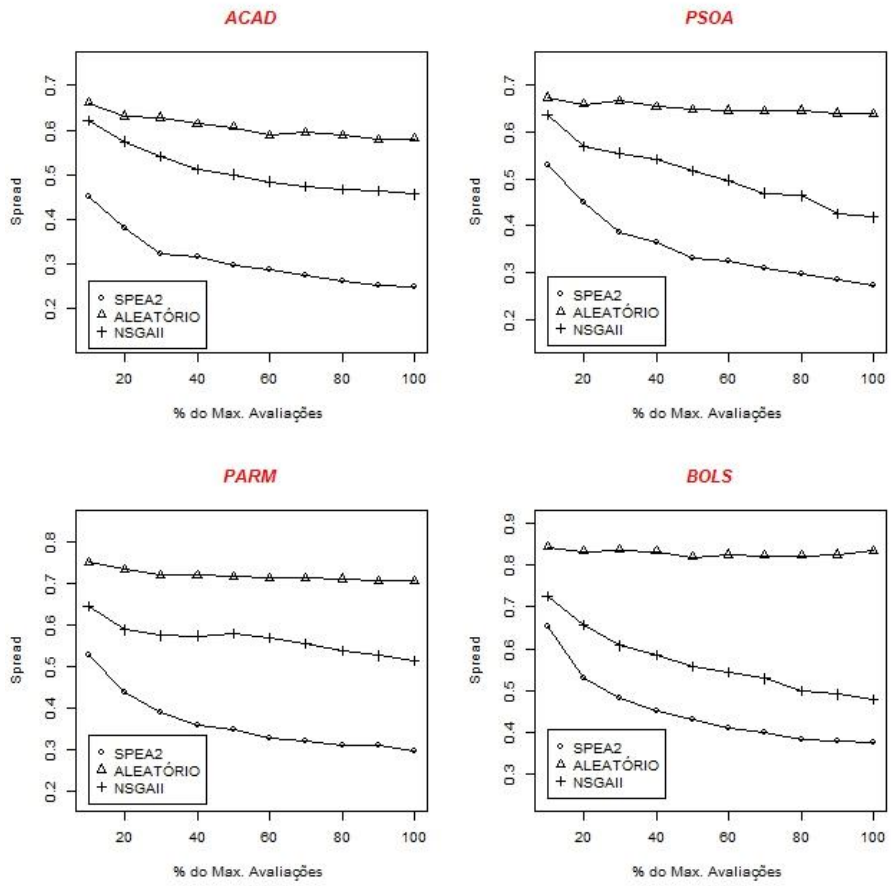

Figura 14. Evolução do *Spread*

Sabendo que quanto menor o *Spread*, melhor é a diversidade das fronteiras de Pareto, uma análise visual da Figura 14 permite verificar para todas as instâncias e número máximo de avaliações que as fronteiras de Pareto encontradas pelo SPEA2 possuem uma melhor diversidade, do ponto de vista do indicador de qualidade *Spread*, quando comparadas às fronteiras encontradas pelos algoritmos NSGA-II e Aleatório. Esta mesma conclusão pode ser feita com a análise da Figura 15 que apresenta quatro gráficos *boxplot* com dados referentes ao número máximo de avaliações igual a (2 x $|T|)^2$ para *Spread.*

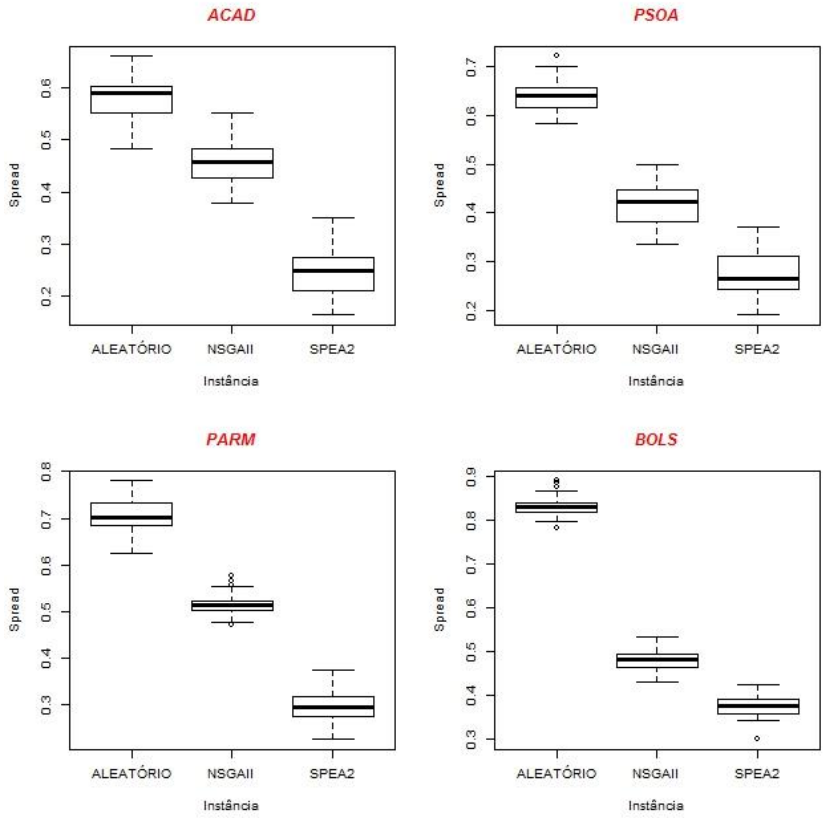

Figura 15. Gráficos *Boxplot* para *Spread*

Como observado nas Figura 14, o SPEA2 encontra melhores fronteiras de Pareto quando comparado ao NSGA-II e o Aleatório. Como forma de comprovar estatisticamente tal observação, a Tabela 16 apresenta os *p-values* e *effect-size* referentes a esta comparação.

Tabela 16. Análise de indicador de qualidade: *Spread*

Instância	Effect-Size			P-value		
	SPEA2>NSGA-II	AL.>SPEA2	AL.>NSGA-II	SPEA2xNSGA-II	AL.xSPEA2	AL.xNSGA-II
ACAD	0%	97%	98%	<0,01	<0,01	<0,01
PSOA	0,01%	100%	100%	<0,01	<0,01	<0,01
PARM	0%	100%	100%	<0,01	<0,01	<0,01
BOLS	0%	100%	100%	<0,01	<0,01	<0,01

As análises feitas a partir da visualização dos gráficos (Figura 14 e Figura 15) podem ser confirmadas pelos valores de *p-values* da Tabela 16. A partir destes valores é possível afirmar com 99% de certeza que as fronteiras de Pareto geradas pelo SPEA2 são diferentes, quanto ao *Spread*, das geradas pelo NSGA-II, e Aleatório, refutando assim a hipótese $\mathbf{H_{0\text{-}\Delta}}$ para todas as instâncias. Sabendo que quanto menor o valor do

Spread, melhor é a diversidade da fronteira de Pareto e, como apresentado na Figura 15, as médias das fronteiras geradas pelo SPEA2 foram menores, é possível aceitar a $\mathbf{H_{1\text{-}\Delta}}$ para todas as instâncias.

Apesar do maior número de soluções nas fronteiras de Pareto geradas pelo NSGA-II, percebe-se a partir da visualização dos gráficos representados na Figura 9, que as soluções geradas por este algoritmo se concentram no meio do espaço de busca, dificilmente encontrando soluções que estão nas extremidades deste espaço. Como visto na Seção 4.3.1, o indicador de qualidade *Spread* considera a distância das soluções desta extremidade. Sendo assim, este pode ser o fator que colabora para a pouca diversidade quando analisada do ponto de vista do *Spread*.

4.3.3.5 Análise do *Hipervolume*

De forma análoga as análises anteriores, a Figura 16 apresenta as médias do *Hipervolume* a medida que o número máximo de avaliações atribuído aos algoritmos aumenta.

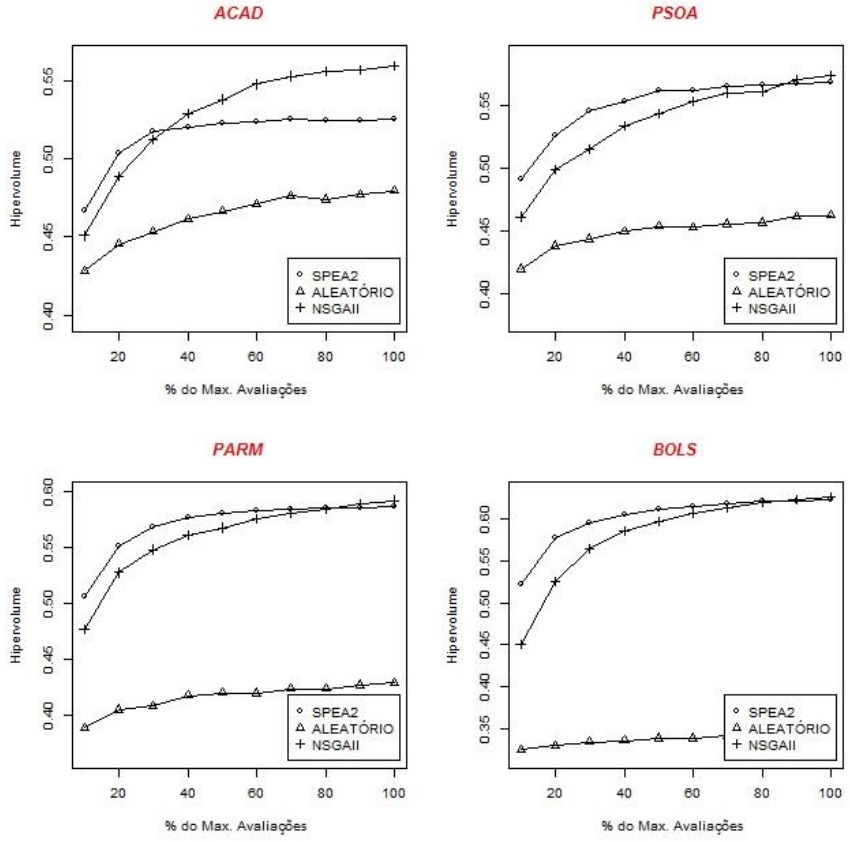

Figura 16. Evolução do *Hipervolume*

Sabendo que quanto maior o *Hipervolume*, melhor é a diversidade e convergências das fronteiras de Pareto geradas, o comportamento dos resultados quanto a este indicador de qualidade é similar ao observado para *Error Ratio* e *Generational Distance*. Nas quatro instâncias, o SPEA2 produz melhores fronteiras de Pareto, quando comparadas ao NSGA-II e ao Aleatório, com número máximo de avaliações menor, e o algoritmo NSGA-II se equipara e ultrapassa este algoritmo à medida que o número máximo de avaliações aumenta. No caso dos gráficos representados na Figura 16, nas três primeiras instâncias, é possível visualizar que o NSGA-II superou o SPEA2 quanto à qualidade das fronteiras de Pareto geradas considerando o *Hipervolume* quando o número de avaliações foi igual a $(2 \times |T|)^2$. Porém, para a instância 4, não há diferença significativa que se possa concluir visualmente qual produziu melhores fronteiras de Pareto.

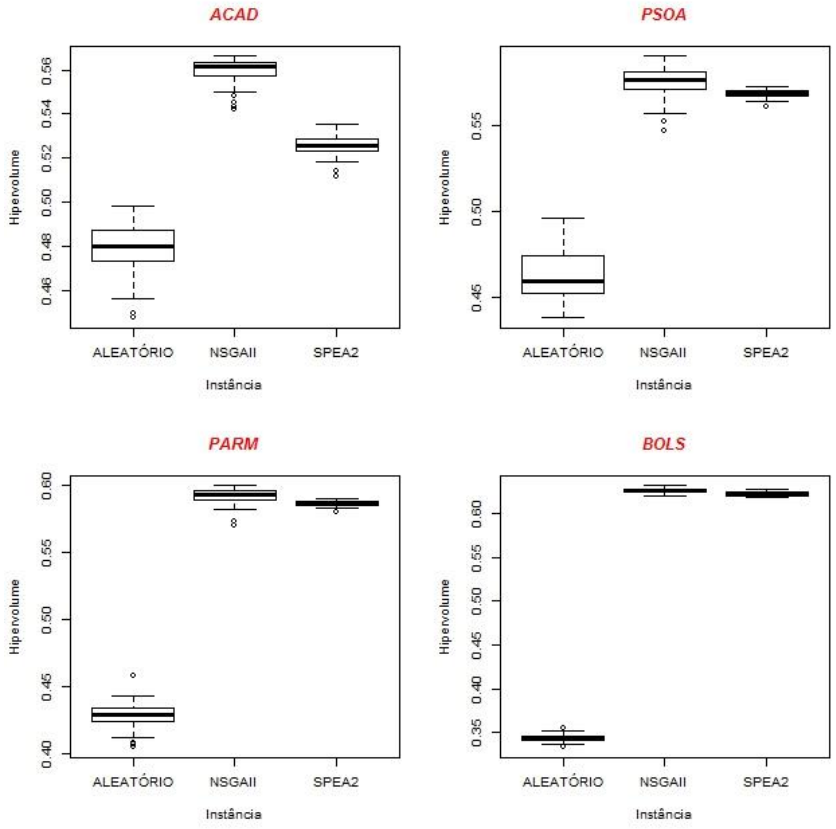

Figura 17. Gráficos *Boxplot* para *Hipervolume*

A Figura 17 mostra que a diferença entre as fronteiras geradas pelos algoritmos foi maior para as três primeiras instâncias quando o número máximo de avaliações foi igual a $(2 \times |T|)^2$. Porém, é possível perceber neste gráfico que existe uma diferença entre o NSGA-II e o SPEA2, onde o NSGA-II passou a produzir melhores fronteiras de Pareto quando comparada ao SPEA2 para este número máximo de avaliações. Para

verificar se esta diferença é estatisticamente significativa, a Tabela 17 apresenta dados de *p-values* e *effect-size*.

Tabela 17. Análise de indicador de qualidade: *Hipervolume*

Instância	Effect-Size			P-value		
	SPEA2>NSGA-II	AL.>SPEA2	AL.>NSGA-II	SPEA2xNSGA-II	AL.xSPEA2	AL.xNSGA-II
ACAD	0%	0%	0%	<0,01	<0,01	<0,01
PSOA	24%	0%	0%	<0,01	<0,01	<0,01
PARM	11%	0%	0%	<0,01	<0,01	<0,01
BOLS	15%	0%	0%	<0,01	<0,01	<0,01

As análises feitas a partir da visualização dos gráficos podem ser confirmadas pela análise dos valores de *p-values* da Tabela 17. A partir destes valores é possível afirmar com 99% de certeza que as fronteiras de Pareto geradas pelo NSGA-II são diferentes, quanto ao *Hipervolume*, das geradas pelo SPEA2 e Aleatório, refutando assim a hipótese $\mathbf{H_{0\text{-}HV}}$ para todas as instâncias. A Figura 16 mostra que as médias do *Hipervolume* das fronteiras geradas pelo NSGA-II foi maior que as geradas pelo SPEA2. Além disso, a partir dos valores do *effect-size*, conclui-se para as quatro instâncias que, na maioria das execuções dos algoritmos, o NSGA-II gerou melhores soluções quando comparado ao SPEA2. Com isto, é possível aceitar a $\mathbf{H_{1\text{-}HV}}$ para todas as instâncias.

4.3.3.6 Análise dos Tempos de Execução dos Algoritmos

A análise do tempo de execução dos algoritmos segue a mesma abordagem adotada para a análise dos indicadores de qualidade. A Figura 18 mostra que o crescimento do tempo de execução é quase linear ao número máximo de avaliações definido. Para as três primeiras instâncias esta diferença é mais nítida, pois a escala dos gráficos é menor. Diferentemente do esperado, o algoritmo Aleatório apresenta um desempenho pior que os outros dois nas duas primeiras instâncias. Para execuções com número máximo de avaliações igual a $(2 \times |T|)^2$, a Figura 19 mostra que o NSGA-II foi mais rápido que o SPEA2 para encontrar as fronteiras de Pareto.

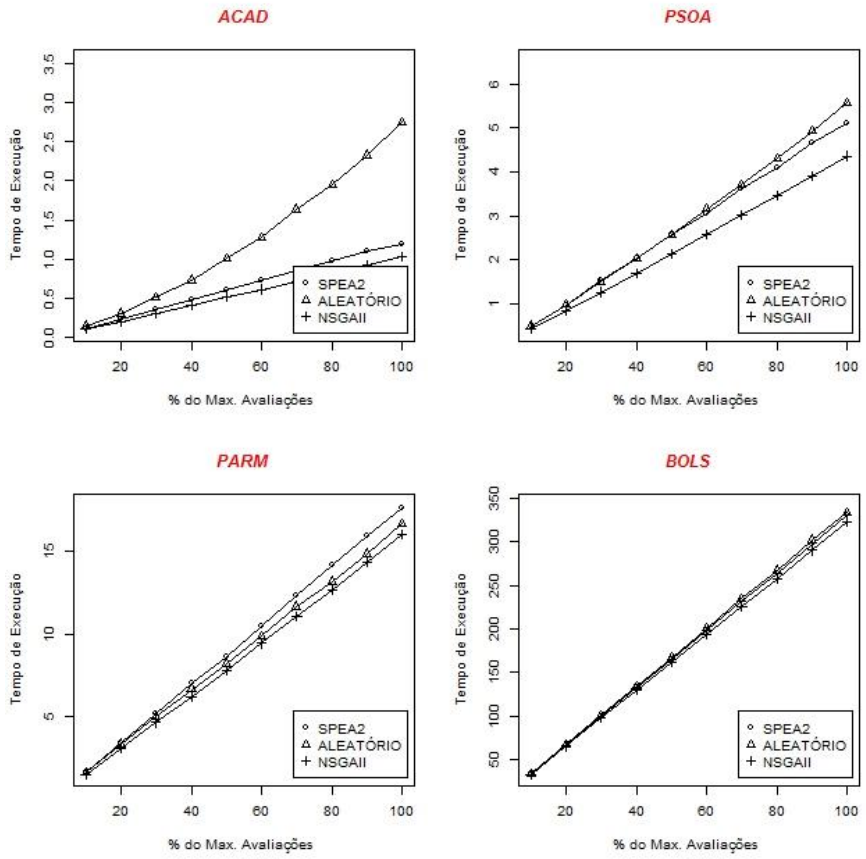

Figura 18. Evolução dos Tempos de Execução

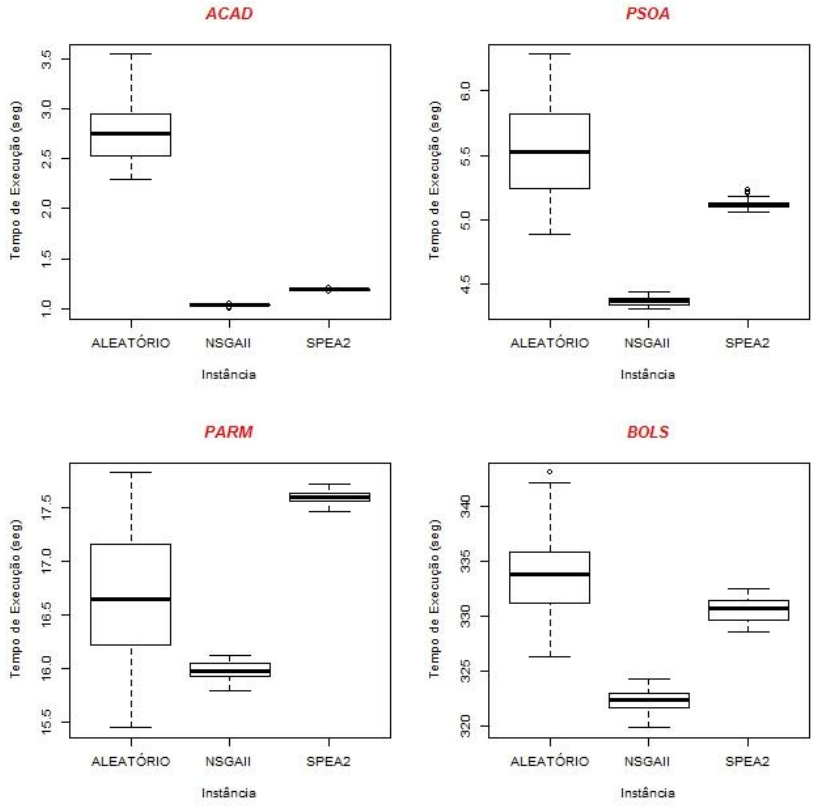

Figura 19. Gráficos *box-plot* para tempo de execução

Para validar as conclusões quanto ao tempo de execução, a Tabela 18 apresenta dados estatísticos de *effect-size* e *p-value* para teste da hipótese $\mathbf{H_{o\text{-}TE}}$.

Tabela 18. Análise dos Tempos de Execução

Instância	Effect-Size			P-value		
	SPEA2>NSGA-II	AL.>SPEA2	AL.>NSGA-II	SPEA2xNSGA-II	AL.xSPEA2	AL.xNSGA-II
ACAD	100%	100%	100%	<0,01	<0,01	<0,01
PSOA	100%	89%	100%	<0,01	<0,01	<0,01
PARM	100%	5%	87%	<0,01	<0,01	<0,01
BOLS	100%	78%	100%	<0,01	<0,01	<0,01

Para todas as instâncias, os valores de *p-value* foram menores do 0,01, permitindo concluir com 99% de certeza que a média do tempo de execução dos algoritmos é diferente, refutando assim a hipótese $\mathbf{H_{o\text{-}TE}}$. Portanto, é possível aceitar a hipótese $\mathbf{H_{1\text{-}TE}}$, já que as médias do tempo de execução do NSGA-II foram menores e o algoritmo foi mais rápido em um maior número de execuções, fato demonstrado pelos valores dos *effect-size*.

4.3.3.7 Considerações Finais a Respeito da Análise Sobre Comparação dos Algoritmos

O objetivo da comparação dos algoritmos, no contexto do NRP-APF multi-objetivo, era avaliar tempo de execução, convergência e diversidade. Quanto ao tempo de execução, foi demonstrado estatisticamente que o NSGA-II possui um desempenho melhor quando comparado ao SPEA2 nas quatro instâncias quando utilizado número máximo de avaliações igual a $(2 \times |T|)^2$. Em uma análise visual, percebeu-se que para a maioria dos cenários (número máximo de avaliações x instâncias) o NSGA-II foi superior aos demais.

Para três dos quatro indicadores de qualidade (*Error Ratio*, *Generational Distance* e *Hipervolume*), conclui-se que o SPEA2 produz melhores resultados quando se deseja obter resultados mais rapidamente, ou seja, com menor número máximo de avaliações. Já o NSGA-II produziu as melhores fronteiras de Pareto com número máximo de avaliações maiores.

Para todas análises estatísticas, foi utilizado o número máximo de avaliações igual a $(2 \times |T|)^2$. Nestas análises verificou-se que, em relação à convergência, a maioria dos indicadores foi favorável ao NSGA-II, ou seja, o NSGA-II produz fronteiras de

Pareto com melhor convergência que as produzidas pelo SPEA2. Como mostrado na seção 4.3.3.2, não foi possível observar esta vantagem para a 4ª instância e o indicador de qualidade *Error Ratio*. No entanto, foi discutido que o tamanho da instância pode ter influenciado e se utilizasse um número máximo de avaliações maior, provavelmente o NSGA-II superaria o SPEA2. Esta comprovação ficará como estudo futuro.

Em relação à diversidade, dois indicadores mostraram-se contraditórios em relação aos resultados obtidos. Enquanto os resultados referentes ao *Spread* indicaram que o SPEA2 produz fronteiras com melhor diversidade, o *Hipervolume* apontou para uma direção contrária, onde o NSGA-II seria melhor quanto à diversidade das soluções ao longo da fronteira. Porém, como discutido na Seção 4.3.3.4, isto pode ser devido ao fato do *Spread* considerar as soluções extremas que não são alcançadas pelo NSGA-II.

4.4 Ameaças à Validade dos Experimentos

De acordo com Wohlin et al. (2000), no planejamento de um projeto experimental é importante se preocupar com a validade do experimento para que os resultados obtidos sejam válidos no contexto assumido. Neste sentido, os autores sugerem avaliar quatro tipos de ameaças à validade, sendo eles: a validade interna, a validade de construção, a validade de conclusão e a validade externa.

Para experimentos com algoritmos de busca no contexto da Engenharia de Software, BARROS e NETO (2011) enumeram, para cada tipo de ameaça, quais são as ameaças que podem comprometer a validade do experimento. Neste trabalho, analisamos os tratamentos dados a cada ameaça, assim como os riscos que podem afetar a validade do experimento. Para isto, esta seção foi dividida em quatro subseções que listam e discutem estas ameaças no contexto dos experimentos deste capítulo.

4.4.1 Ameaças à Validade de Conclusão

Para ameaças à validade de conclusão, BARROS e NETO (2011) enumeram quatro ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

Não consideração da variação randômica. A inicialização de um algoritmo heurístico geralmente é feita de forma randômica. Por exemplo, a população inicial de um algoritmo genético pode ser gerada aleatoriamente. Por isto, os resultados são diferentes em cada execução, podendo uma execução isolada ser muito positiva ou muito negativa.

Para atenuar o efeito de uma execução isolada, sugere-se executar os algoritmos diversas vezes de modo a eliminar esta ameaça. Como descrito nas Seções 4.2.2 e 4.3.2, planejamos que os algoritmos fossem executados 50 vezes.

Falta de boa estatística descritiva. A execução repetida dos algoritmos produz uma amostra de dados relativamente grande. Portanto, o uso de estatísticas descritivas e análises gráficas devem ser recursos bastante explorados na análise do experimento. No presente trabalho, os experimentos utilizaram tabelas e\ou gráficos (de linha e *box-plot*) que apresentavam médias, mínimos, máximos e desvios padrões.

Falta de base comparativa significativa. Estudos que envolvem comparação de algoritmos usualmente analisam os resultados da execução destes algoritmos para as mesmas instâncias. Neste caso, alguns trabalhos utilizam de busca aleatória para mostrar que um determinado problema não pode ser resolvido encontrando soluções randômicas. No entanto, para tirar conclusões confiáveis baseadas em tal comparação, a base para esta comparação deve ser representativa de uma solução de qualidade. No experimento para avaliação do NRP-APF, utilizamos o NRP-CLS que serve de base para diversas extensões anteriores do NRP (FREITAS et al., 2011; XUAN et al., 2012; ZHANG et al., 2007, 2013). No experimento para comparação de algoritmos, utilizamos dois algoritmos genéticos sistemáticos, que são amplamente utilizados no contexto de otimização na área da Engenharia de Software. Neste experimento, também executamos e analisamos o algoritmo aleatório, que serviu como base para validação do experimento.

Falta de formalização de hipóteses e testes estatísticos. Os estudos baseados em comparação devem ser baseados em hipóteses formais que devem ser avaliadas por testes estatísticos apropriados. Testes estatísticos devem ser conduzidos através de procedimentos de inferência estatística paramétricas ou não-paramétricas, de acordo com a distribuição observada nos dados. No presente trabalho, os experimentos foram planejados para realização de testes de hipóteses. Como nem todas as amostras comparadas tinham distribuição normal, utilizamos testes não-paramétricos para a avaliação das hipóteses.

4.4.2 Ameaças à Validade Interna

Para ameaças à validade interna, BARROS e NETO (2011) enumeram quatro ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

Parametrização dos algoritmos não são considerados. Os parâmetros utilizados na comparação de uma nova proposta ou comparação de algoritmos não são explicitamente apresentados no projeto experimental. Ao esconder os ajustes de parâmetros, pode-se favorecer uma ou outra técnica, limitando a generalização das conclusões observadas. Além disso, uma descrição completa de valores dos parâmetros é necessária para o experimento ser passível de reprodução. Apesar do presente trabalho não incluir um estudo para calibração dos parâmetros a serem usados pelos algoritmos, apresentamos todos os parâmetros utilizados por estes algoritmos. Além disto, utilizamos como base o trabalho de DURILLO et al. (2009) que utiliza os mesmos parâmetros para execução de algoritmos genéticos no contexto do NRP.

Falta de discussão sobre a implementação dos algoritmos. O código-fonte usado em um experimento pode esconder ajustes ou instrumentações específicas para favorecer certos casos ou algoritmos, influenciando assim os resultados observados. O código fonte deve ser tornado público, permitindo que outros pesquisadores reproduzam e fiscalizem o experimento. Nos dois experimentos utilizamos o *JMetal* que disponibiliza seu código-fonte para o público. Para o experimento com objetivo de comparar os algoritmos, a ameaça A6 tem impacto, pois a implementação da proposta é compartilhada entre os algoritmos, que possui código disponibilizado. Para o experimento com objetivo de avaliar o NRP-APF, os códigos das implementações das propostas NRP-APF e NRP-APF estão disponíveis em https://github.com/vitorpadilha/otimizacao-pontos-de-funcao.

Falta de clareza na descrição dos procedimentos e ferramentas de coleta de dados. Os procedimentos e ferramentas utilizados para coleta de informações de instâncias do mundo real ou aleatórias devem ser descritos com precisão para se certificar de que aspectos descritos não influenciem os resultados do estudo, beneficiando casos favoráveis ao pesquisador. Na Seção 4.1 descrevemos as características das instâncias utilizadas nos estudos e todos os tratamentos aplicados sobre as instâncias reais. Também discutimos que transformação foi aplicada para que as instâncias do NRP-APF fossem utilizadas no NRP-CLS. Também discutimos, na Seção 4.2.4, como a instância ACAD foi alterada para analisarmos a relação entre tamanhos de função de dados e os resultados produzidos pelo NRP-APF.

Falta de instâncias de problemas reais. Engenharia de Software é uma ciência prática e, como tal, deve lidar com problemas reais. Embora casos gerados aleatoriamente

possam ser úteis para tratar o comportamento de uma nova técnica, em certas situações estes casos podem não ter as propriedades encontradas em casos reais. Com isto, podem influenciar as conclusões tiradas a partir do experimento. Os dois experimentos apresentados neste capítulo utilizaram instâncias reais, apesar de terem sido manipuladas para adequação ao problema.

4.4.3 Ameaças à Validade de Construção

Para o cenário de validade de construção, BARROS e NETO (2011) enumeram três ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

Falta de medidas para avaliar custo de execução. Ao medir o custo de execução de um algoritmo, o pesquisador deve justificar a métrica selecionada. Segundo BARROS e NETO (2011), o número de avaliações da função de *fitness* é a medida mais aceita e usada para medir custo de execução de algoritmos. Nos dois experimentos utilizamos a mesma métrica para controlar o tempo de execução dos algoritmos: um número máximo de avaliações da função de *fitness* previamente determinado.

Falta de medidas que avaliem com qualidade dos resultados. As medidas escolhidas devem ser relevantes para o problema: uma melhora em seus valores deve estar relacionada com a melhoria da qualidade da solução. Portanto, a seleção de medidas adequadas é relevante para se certificar de que os resultados observados aceitam ou rejeitam a teoria proposta. No experimento para avaliação do NRP-APF mono-objetivo, utilizamos o grau de satisfação dos patrocinadores como métrica para avaliar a qualidade das soluções. Tal métrica foi utilizada em trabalhos anteriores (DEL SAGRADO et al., 2011; FREITAS et al., 2011). Já para o experimento com objetivo de comparar os algoritmos, diversificamos a análise utilizando quatro indicadores de qualidade (vide Seção 4.3.1) propostos na literatura a fim de avaliar a qualidade das fronteiras de Pareto geradas por estes algoritmos do ponto de vista de convergência e diversidade.

Não discutir se o modelo proposto reflete o mundo real. O ambiente no qual o software é desenvolvido geralmente é complexo, podendo envolver questões técnicas e sociais. Portanto, qualquer modelo que descreve um problema relacionado ao software é uma simplificação do mundo real. Ao formalizar esses modelos, é preciso discutir suas limitações e como as simplificações adotadas podem influenciar na prática. Neste

trabalho, não discutimos a validade do NRP por se basear em uma formalização apresentada em um trabalho anterior. Somente adicionamos uma técnica de estimativa (Análise de Pontos de Função) ao problema. A APF é uma técnica exata, utilizada por empresas de desenvolvimento de software.

4.4.4 Ameaças à Validade Externa

Para o cenário de validade externa, BARROS e NETO (2011) enumeram três ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

Falta de definição clara das instâncias. Nenhuma generalização é possível se os pesquisadores não podem conhecem as instâncias utilizadas em um estudo empírico. Portanto, qualquer projeto experimental deve fornecer uma definição clara das instâncias selecionadas. Na Seção 4.1, discutimos todos os aspectos das instâncias utilizadas nos experimentos.

Falta de uma estratégia objetivo de seleção de objetos. O experimento deverá mostrar claramente como as instâncias utilizadas no experimento foram selecionadas, projetadas, geradas aleatoriamente, ou capturadas de problemas do mundo real. O pesquisador deve justificar como estas instâncias são usadas no experimento em questão. Na Seção 4.1, discutimos todos os aspectos das instâncias utilizadas nos experimentos, entre eles a razão pela qual estas instâncias foram selecionadas.

Falta de diversidade de tamanho e complexidade nas instâncias. Técnicas de Engenharia de Software são projetadas para lidar com sistemas e equipes que podem variar em tamanho e complexidade. Portanto, um experimento deve avaliar uma técnica em uma amplitude de instâncias de problemas, variando em tamanho e complexidade, para fornecer uma avaliação sobre os limites da nova proposta. As quatro instâncias utilizadas nesta dissertação possuem características bem distintas, principalmente em relação ao tamanho funcional. O número de transações, por exemplo, está entre o intervalo de 39 a 200 funções de transação. No entanto, a variação do número de patrocinadores das instâncias foi pequena, sendo que três das quatro instâncias tinham exatamente o mesmo número de patrocinadores. Esta limitação pode provocar restrições na capacidade de generalizar os resultados dos estudos.

4.5 Considerações Finais

Este capítulo apresentou os resultados dos experimentos realizados para avaliar o NRP-APF em relação ao NRP-CLS e verificar qual algoritmo produz melhores soluções no contexto do NRP-APF.

Para o experimento com objetivo de avaliar o NRP-APF, duas análises foram feitas. A primeira comparou as soluções geradas pelas propostas NRP-APF e NRP-CLS. Uma análise de estatística com teste não paramétrico mostrou que o NRP-APF produziu melhores soluções quando comparado ao NRP-CLS, para alguns dos cenários apresentados na seção 4.2.2. A segunda análise teve o objetivo de caracterizar e relacionar o comportamento do NRP-APF com os tamanhos das funções de dados de uma instância de software. Esta análise o NRP-APF possui maior vantagem quando uma instância de software possui funções de dados mais agrupadas, ou seja, que contêm mais DER e RLR e, consequentemente, maior complexidade.

A análise do experimento com objetivo de comparar os algoritmos mostrou que, para três dos quatro indicadores de qualidade (*Error Ratio*, *Generational Distance* e *Hipervolume*), o SPEA2 produziu melhores resultados quando se deseja obter soluções em um tempo menor, ou seja, com menores números máximos de avaliações. Já o NSGAII produziu as melhores fronteiras de Pareto com número máximo de avaliações maiores. Nos testes de hipóteses para indicadores de qualidade relacionados à convergência das fronteiras de Pareto (*Error Ratio*, *Generational Distance* e *Hipervolume*), a maioria dos indicadores foi favorável ao NSGAII, ou seja, o NSGAII produziu fronteiras de Pareto com melhor convergência das produzidas pelo SPEA2. Como mostrado na Seção 4.3.3.2, não foi possível observar esta vantagem quando usada a 4ª instância para o indicador de qualidade *Error Ratio*. No entanto, foi discutido que o tamanho da instância pode ter influenciado e se utilizasse um número máximo de avaliações maior, provavelmente o NSGAII superaria o SPEA2.

Em relação à diversidade, a análise de dois indicadores de qualidade mostrou-se contraditória. Enquanto os resultados referentes ao *Spread* indicaram que o SPEA2 produz fronteiras com melhor diversidade, o *Hipervolume* apontou para uma direção contrária, onde o NSGAII produz fronteiras com melhor diversidade. Porém, com discutido na Seção 4.3.3.4, isto pode ser ao fato do *Spread* considerar as soluções extremas que não são bem alcançadas pelo NSGAII, fato demonstrado na Figura 9.

O próximo capítulo apresenta as contribuições, limitações do presente trabalho e perspectivas de melhoria futura.

Capítulo 5 - Conclusões

5.1 Considerações Finais e Contribuições

Neste trabalho, foi apresentada uma nova proposta para o NRP com objetivo de torná-lo um modelo mais prático, pois foi utilizada a técnica de APF, bastante utilizada pelo mercado de desenvolvimento de software, como base para formalização do problema. Tratando-se de um problema de análise combinatória NP-Completo, o NRP pode fazer uso de heurísticas para encontrar soluções boas dentro de um tempo razoável. Após a apresentação da nova proposta, o NRP-AFP, uma avaliação foi feita a fim de medir a qualidade das soluções geradas por esta proposta, bem como um estudo inicial com objetivo de verificar o comportamento de três algoritmos quando executados no contexto do NRP-APF.

O estudo para avaliação do NRP-APF utilizou de quatro instâncias reais e baseou-se na comparação do NRP-APF mono-objetivo com o NRP-CLS com objetivo de verificar qual das propostas gerava soluções que melhor atendiam aos patrocinadores do projeto. Este estudo mostrou que o NRP-APF é estatisticamente melhor que o NRP-CLS para alguns cenários de instâncias e esforço disponível. Um detalhamento deste estudo, com o objetivo de relacionar a eficiência do NRP-APF com o tamanho das funções de dados da instância, mostrou que quanto maior a proporção de funções de dados de média e alta complexidade da instância, maior será a vantagem do NRP-APF em relação ao NRP-CLS.

Neste sentido, o estudo mostrou a criação de uma proposta ao NRP na qual o esforço de desenvolvimento dos requisitos varia de acordo com outros requisitos do software (NRP-APF), produzindo melhores soluções quando comparado a proposta onde os requisitos são atômicos (NRP-CLS). A explicação é que, no caso da NRP-APF, a função de dados pode ter o esforço para desenvolvimento (em PF) diminuído, de acordo com as funções de transação selecionadas, enquanto na NRP-CLS este esforço é invariável para todos os requisitos. Portanto, este trabalho contribuiu com uma nova proposta, que se mostrou eficaz para a solução do NRP, principalmente quando

comparada à proposta clássica. Além disto, a proposta utiliza da APF, técnica bastante utilizada pelo mercado de software, sendo assim facilmente aplicada ao processo de desenvolvimento de software das empresas que já utilizam esta técnica. Com isto, empresas que adotarem o modelo proposto, atenderão as expectativas dos patrocinadores nas primeiras *releases*, recebendo assim maior incentivo para terminar as demais e para projetos futuros.

Uma segunda análise dos resultados com objetivo de relacionar o tamanho das funções de dados de um software com a eficácia do NRP-APF, mostrou que o NRP-APF é mais eficaz que o NRP-CLS para instancias de software que possuem funções de dados mais agrupadas. Isto porque é possível diminuir o esforço para construção das funções de dados de média ou alta complexidade. Este esforço é diminuído quando elementos (DER e\ou RLR) de uma função de dados não são usados por nenhuma função de transação que será considerada para a *release*. Deste modo, esta função de dados pode ter sua complexidade diminuída e, consequentemente, o esforço para desenvolvimento também. Com a diminuição do esforço para desenvolvimento de uma ou mais função de dados, que só é possível no NRP-APF (vide Seção 4.1), cria-se um espaço, é possível encaixar novas funções de transação que estariam fora do escopo, aumentando assim a satisfação dos patrocinadores do projeto. Com base nisto, uma empresa contratante de serviços de desenvolvimento de software pode utilizar esta técnica para levantamento de requisitos e negociação de contrato a fim de obter vantagens econômicas. Por exemplo, um cenário que seja possível agrupar funções de dados relacionadas, permitiria gerar melhor satisfação dos patrocinadores do software nas primeiras entregas quando comparado a um cenário em que não há este agrupamento.

Com objetivo de caracterizar o comportamento de três algoritmos multi-objetivo no contexto do NRP-APF bi-objetivo, um segundo estudo foi conduzido com as mesmas instâncias da primeira avaliação da proposta. Os algoritmos avaliados foram o NSGA-II, o SPEA2 e o Aleatório. Para avaliação dos algoritmos, o estudo utilizou de quatro indicadores de qualidade (*Error Ratio*, *Generational Distance*, *Spread* e *Hipervolume*) como métricas para medir a qualidade das soluções geradas pelos algoritmos e o tempo de execução para avaliar o desempenho. Um resumo dos resultados obtidos a partir destes indicadores de qualidade mostrou que, com poucas avaliações (ou seja, pouco tempo de execução), o algoritmo SPEA2 gera soluções melhores que o NSGA-II. No entanto, a medida que o número de avaliações aumenta, o

NSGA-II passa a produzir melhores soluções quando comparado ao SPEA2. O Algoritmo Aleatório neste estudo serviu apenas como "teste de sanidade", como medida de verificar se o experimento estava sendo conduzido de maneira correta, pois é um algoritmo que não produz bons resultados e não poderia ser melhor que os outros. Com relação ao tempo de execução, o estudo mostrou que o algoritmo Aleatório apresentou o pior desempenho. O NSGA-II foi o que executou as mesmas avaliações em menor tempo. Portanto, foi o que apresentou melhor desempenho.

A comparação dos algoritmos confirma o bom desempenho apresentado pelo NSGA-II no contexto do NRP. Porém, este trabalho foi além e mostrou que para um tempo menor ele pode ser menos eficiente quando comparado ao SPEA2. A fim de preservar a validade dos experimentos algumas medidas foram adotadas, como: (a) utilizar ferramentas conhecidas e testadas por trabalhos anteriores, como *JMetal*; (b) usar instâncias reais e heterogêneas, além de detalhar as alterações necessárias para adequação à formalização do problema; (c) gerar um grande volume de amostras que permite maior confiabilidade dos testes estatísticos; e (d) usar de métricas e indicadores de qualidade conhecidos e utilizados em trabalhos anteriores.

5.2 Limitações e perspectivas futuras do trabalho

Em relação à formulação do problema existem dois possíveis aprimoramentos. O primeiro é transformar funções de dados que são do tipo ALI em AIE, quando somente transações de saída e consulta (SE e CE) utilizam estas funções de dados. O segundo é tornar a formulação do problema iterativa, de maneira que os patrocinadores possam tomar decisões no momento em que a busca heurística está sendo executada. Um exemplo de decisão seria: "A função de dados d_i (tipo ALI) apresenta 2 ALR e 5 DER, com complexidade Média, deseja retirar um DER para torná-la de baixa complexidade? Se sim, qual?".

Em relação à avaliação do NRP-APF, espera-se que outros estudos sejam feitos com maior número de instâncias e utilizando outras propostas ligadas ao NRP como base para comparação. Além disto, a avaliação foi feita apenas baseando-se na formalização mono-objetiva. Espera-se, no futuro, que estudos também avalie a formalização bi-objetiva comparando com a formalização bi-objetiva clássica, apresentada por DURILLO et al. (2009).

Quanto à comparação dos algoritmos, foram utilizados apenas os algoritmos multi-objetivo NSGA-II, SPEA2 e o Aleatório. Em trabalhos futuros a ideia é que

outros algoritmos possam ser usados e analisados. Além disso, deve-se explorar a comparação dos algoritmos mono-objetivo no contexto do NRP-APF mono-objetivo, pois apenas o algoritmo genético generacional foi utilizado no primeiro experimento.

Por fim, a proposta do NRP-APF é simples de ser aplicada em empresas que já utilizam a APF como técnica para medição de esforço e espera-se que estudos sejam feitos em ambiente real a fim de medir a real satisfação de patrocinadores de projetos de software.

Referências

AHN, Y.; SUH, J.; KIM, S.; KIM, H. "The software maintenance project effort estimation model based on function points", *Journal of Software Maintenance and Evolution: Research and Practice*, v. 15, n. 2, p. 71–85, 2003.

BAGNALL, A. J.; RAYWARD-SMITH, V. J.; WHITTLEY, I. M. "The next release problem", *Information and Software Technology*, v. 43, n. 14, p. 883–890, 2001.

BARROS, M. DE O. "An Analysis of the Effects of Composite Objectives in Multiobjective Software Module Clustering", *In: Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference*. p. 1205–1212, New York, NY, USA ACM, 2012. Disponível em: <http://doi.acm.org/10.1145/2330163.2330330>

BARROS, M. DE O.; NETO, A. C. D. *Threats to Validity in Search-based Software Engineering Empirical Studies*, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2011.

BOEHM, B. W.; ABTS, C.; BROWN, A. W.; CHULANI, S.; CLARK, B. K.; HOROWITZ, E.; MADACHY, R.; REIFER, D. J.; STEECE, B. *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000.

BRASIL. "Guia Prático para Contratação de Soluções de Tecnologia da Informação versão 1.1", Brasília: SLTI (Secretaria de Logística e Tecnologia da Informação) /MPOG, 2011.

COSMIC. *COSMIC-FFP Measurement Manual: Versão 2.2*, Common Software Measurement International Consortium, 2003.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. "A fast and elitist multiobjective genetic algorithm: NSGA-II", *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 2, p. 182–197, 2002.

DEL SAGRADO, J.; ÁGUILA, I. M. "Ant Colony Optimization for requirement selection in incremental software development", *Symposioum on Search Based Software*, Windsor, UK, 2009.

DEL SAGRADO, J.; ÁGUILA, I. M.; ORELLANA, F. J. "Requirements interaction in the next release problem", *In: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation*, p. 241–242, Dublin, Ireland, 2011.

DURILLO, J. J.; NEBRO, A. J.; ALBA, E. “The jMetal framework for multi-objective optimization: Design and architecture.” In: *Evolutionary Computation CEC 2010 IEEE Congress on*. v. 5467, p. 18–23. IEEE, 2010. Disponível em: <http://www.lcc.uma.es/~antonio/investigacion/articulos/cec2010.pdf>

DURILLO, J. J.; ZHANG, Y.; ALBA, E.; HARMAN, M.; NEBRO, A. J. "*A study of the bi-objective next release problem"*, Empirical Software Engineering, v. 16, n. 1, p. 29–60, 2011.

DURILLO, J. J.; ZHANG, Y.; ALBA, E.; NEBRO, A. J. “A Study of the Multi-objective Next Release Problem”, *1st International Symposium on Search Based Software Engineering*, p. 49–58, Windsor, UK, 2009.

FERREIRA, R. C. DA C.; HAZAN, C. “Uma Aplicação da Análise de Pontos de Função no Planejamento e Auditoria de Custos de Projetos de Desenvolvimento de Sistemas”, *Workshop de Computação Aplicada em Governo Eletrônico*, jul. 2010.

FREITAS, F. G.; COUTINHO, D. P.; SOUZA, J. T. DE. “Software Next Release Planning Approach through Exact Optimization”, *International Journal of Computer Applications*, v. 22, n. 8, p. 1–8, 2011.

GONÇALVES, V. P.; BARROS, M. DE O. “Modelando o Problema da Próxima Release sob a Perspectiva da Análise de Pontos de Função.” *IV Worshop de Engenharia de Software Baseada em Busca*, v. 01, p. 12–21, 2013.

GREER, D.; RUHE, G. “Software release planning: an evolutionary and iterative approach”, *Information and Software Technology*, v. 46, n. 4, p. 243–253, 2004.

IFPUG. *Function Point Counting Practices Manual (Release 4.3), 2009*.

JIANG, H.; ZHANG, J.; XUAN, J.; REN, Z.; HU, Y. “A Hybrid ACO algorithm for the Next Release Problem.” *In: Software Engineering and Data Mining SEDM 2010 2nd International Conference on. Chengdu, China 2010*.

KNOWLES, J.; CORNE, D. “The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation”, *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, v. 1, Washington, DC, USA, 1999.

LI, C.; AKKER, M.; BRINKKEMPER, S.; DIEPEN, G. “An integrated approach for requirement selection and scheduling in software release planning”, *Requirements Engineering*, v. 15, n. 4, p. 375–396, 2010.

LI, C.; AKKER, J. M. VAN DEN; BRINKKEMPER, S.; DIEPEN, G. “Integrated requirement selection and scheduling for the release planning of a software product.” *In: Proceeding of the 13th International Workshop on Requirements Engineering Foundation for Software Quality REFSQ 2007*, p. 93–108, Trondheim, Norway, 2007.

LIM, S.; HARMAN, M.; SUSI, A. “Using Genetic Algorithms to Search for Key Stakeholders in Large-Scale Software Projects”, *Aligning Enterprise, System, and Software Architectures*, p. 118-134, 2012.

MATSON, J. E.; BARRETT, B. E.; MELLICHAMP, J. M. “Software development cost estimation using function points”, *IEEE Transactions on Software Engineering*, v. 20, n. 4, p. 275 –287, abr. 1994.

NEBRO, A. J.; DURILLO, J. J.; LUNA, F.; DORRONSORO, B.; ALBA, E. “MOCell: A cellular genetic algorithm for multiobjective optimization”, *International Journal of Intelligent Systems*, v. 24, n. 7, p. 726–746, 2009.

PITANGUEIRA, A. M.; MACIEL, R. S. P.; OLIVEIRA BARROS, M.; ANDRADE, A. S. “A Systematic Review of Software Requirements Selection and Prioritization Using SBSE Approaches”, *Search Based Software Engineering*, Lecture Notes in Computer Science. v. 8084, p. 188–208, 2013.

REEVES, C. R. (ED.). “Modern heuristic techniques for combinatorial problems”, McGraw-Hill, 1995.

SOUZA, J. T. DE; MAIA, C. L. B.; FERREIRA, T. DO N.; CARMO, R. A. F. DO; BRASIL, M. M. A. “An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements”, In: *Search Based Software Engineering*. Springer Berlin Heidelberg, p. 142–157, 2011.

TONELLA, P.; SUSI, A.; PALMA, F. “Using Interactive GA for Requirements Prioritization” *In: Second International Symposium on Search Based Software Engineering (SSBSE)*, v. 2nd, n. Section II, p. 57–66. 2010.

TONELLA, P.; SUSI, A.; PALMA, F. “Interactive requirements prioritization using a genetic algorithm”, *Information and Software Technology*, v. 55, n. 1, p. 173–187, jan. 2013.

VARGHA, A.; DELANEY, H. D. “A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong”, *Journal of Educational and Behavioral Statistics*, v. 25, n. 2, p. 101–132, 2000.

VELDHUIZEN, D. A. VAN. “Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations”, Wright Patterson AFB, OH, USA: Air Force Institute of Technology, 1999.

VELDHUIZEN, D. A. VAN; LAMONT, G. B. “Multiobjective Evolutionary Algorithm Research: A History and Analysis”, Wright Patterson AFB, OH, USA: Air Force Institute of Technology, 1998.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering: an introduction*, Kluwer Academic Publishers, v. 15, 2000.

XUAN, J.; JIANG, H.; REN, Z.; LUO, Z. “Solving the Large Scale Next Release Problem with a Backbone-Based Multilevel Algorithm”, *IEEE Transactions on Software Engineering*, v. 38, n. 5, p. 1195–1212, 2012.

ZHANG, Y.; HARMAN, M. “Search Based Optimization of Requirements Interaction Management.” *In: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*, Benevento, Italy, 2010.

ZHANG, Y.; HARMAN, M.; FINKELSTEIN, A.; AFSHIN MANSOURI, S. “Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation”, *Information and Software Technology*, v. 53, n. 7, p. 761–773, jul. 2011.

ZHANG, Y.; HARMAN, M.; LIM, S. L. “Empirical evaluation of search based requirements interaction management”, *Information and Software Technology*, v. 55, n. 1, p. 126–152, jan. 2013.

ZHANG, Y.; HARMAN, M.; MANSOURI, S. A. “The multi-objective next release problem”, *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO ’07*, p. 1129–1136, New York, New York, USA, 2007.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. “SPEA2: Improving the Strength Pareto Evolutionary Algorithm”, *In: Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. p. 95–100. Zurich, Switzerland International Center for Numerical Methods in Engineering, 2001.