UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS WINTHER DE ALCANTARA ROQUE

ALOCAÇÃO DE RECURSOS HUMANOS EM PROJETOS ÁGEIS DE SOFTWARE BASEADA EM SIMILARIDADE DE TAREFAS

FORTALEZA – CEARÁ

2016

LUCAS WINTHER DE ALCANTARA ROQUE

ALOCAÇÃO DE RECURSOS HUMANOS EM PROJETOS ÁGEIS DE SOFTWARE BASEADA EM SIMILARIDADE DE TAREFAS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências e Tecnologia da Universidade Estadual do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. PhD. Jerffeson Teixeira de Souza

FORTALEZA – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação

Universidade Estadual do Ceará

Sistema de Bibliotecas

Roque, Lucas Winther de Alcantara Roque.
Alocação de Recursos Humanos em projetos ágeis de software baseada em similaridade de tarefas [recurso eletrônico] / Lucas Winther de Alcantara Roque Roque. - 2017.
1 CD-ROM: il.; 4 ¾ pol.

CD-ROM contendo o arquivo no formato PDF do trabalho acadêmico com 60 folhas, acondicionado em caixa de DVD Slim (19 x 14 cm x 7 mm).

Dissertação (mestrado acadêmico) - Universidade Estadual do Ceará, Centro de Ciências e Tecnologia, Mestrado Acadêmico em Ciência da Computação, Fortaleza, 2017.
Área de concentração: Ciência da Computação.
Orientação: Prof. Dr. Jerffeson Teixeira de Souza.

1. Alocação de recurso humanos. 2. Otimização multiobjetivo. 3. Similaridade de tarefas. I. Título.

ATA DA NONAGÉSIMA SEXTA DEFESA PÚBLICA DE DISSERTAÇÃO DE MESTRADO

Aos segundo dia do mês de fevereiro de dois mil e dezessete, no miniauditório do prédio de Pesquisa e Pós-Graduação em Computação, do Mestrado Acadêmico em Ciência da Computação – MACC, realizou-se a sessão pública de defesa da dissertação de **Lucas Winther de Alcantara Roque** aluno regularmente matriculado no Mestrado Acadêmico em Ciência da Computação–MACC, intitulada: **"ALOCAÇÃO DE RECURSOS HUMANOS EM PROJETOS DE SOFTWARE ÁGEIS BASEADO EM SIMILIARIDADE DE TAREFAS"**. A Banca Examinadora reuniu-se no horário de 15:00h às 17:00 horas, sendo constituída pelos Professores Doutores Jerffeson Teixeira de Souza (Orientador/UECE), Paulo Henrique Mendes (UECE) e Tibérius de Oliveira e Bonates (UFC). Inicialmente o mestrando expôs seu trabalho e a seguir foi submetido à arguição pelos membros da Banca, dispondo cada membro de tempo para tal. Finalmente a Banca reuniu-se em separado e concluiu por considerar o mestrando aprovado, por sua dissertação e sua defesa pública. Eu, **Prof.**, Orientador e Presidente da Banca, lavrei a presente ata que será assinada por mim e demais membros da Banca. Fortaleza, 02 de fevereiro de 2017.

PhD. Jerffeson Teixeira de Souza
(Orientador/UECE)

Paulo Henrique Mendes
(UECE)

Tibérius de Oliveira e Bonates
(UFC)

À minha esposa, pela compreensão e companheirismo. À minha família, pelo exemplo.

AGRADECIMENTOS

Primeiramente a Deus e por suas misericórdias, por tudo que tem feito durante toda minha vida. A Ele toda honra, a Ele toda glória.

A minha esposa Isi, por sua compreensão e pelas noites que passou em claro ao meu lado. Por estar do meu lado não apenas neste desafio, mas também no ainda maior que estar por vir.

Aos meu pai Denilson, minha mão Dalvinha e meu Irmão Matheus, por todo o amor, suporte e todos os esforços que nunca foram medidos quando se tratou da minha educação. Em especial a meu pai, exemplo de homem, pai e marido.

Agradeço ao meu mais antigo amigo Heitor Freire, aos amigos Hitalo Matheus, Caio César e Paulo Menezes, que apesar da distância estiveram sempre próximos e torcendo por mim.

Aos professores Roberto Michelan e Marcos Duarte, que me fizeram vislumbrar a carreira acadêmica

Ao Mestrado Acadêmico em Ciência da Computação (MACC), seu corpo docente, direção e administração por todos os ensinamentos que de maneira nenhuma serão tirados de mim. À FUNCAP pelo fomento.

Agradeço aos amigos Italo Yeltsin, Duany Dreyton, Denis Caminha do Grupo de Otimização em Engenharia de Software (GOES.UECE), pela amizade que sempre carregarei comigo. Em especial ao Allysson Alex e Altino Dantas pelas orientações e cooperação, e ao Raphael Saraiva pelo excelente apoio técnico. Agradeço ao Thiago Nascimento, por ter sido meu primeiro incentivador e exemplo de cientista.

Ao meu orientador professor Jerffeson Teixeira que me deu essa oportunidade e confiou em mim. Pelas orientações e ensinos técnicos, mas principalmente pelos ensinos de vida. Muito Obrigado.

Ao Prof. PhD. Paulo Henrique Mendes Maia que contribui com meu aprendizado nas disciplinas, e com minha pesquisa desde a qualificação e ao Prof. PhD. Tibérius de Oliveira e Bonates, por aceitar o convite de participação e contribuição na minha banca avaliadora.

"Caminhamos pela fé, e não pela visão."

(Paulo de Tarso)

RESUMO

A Engenharia de Software (ES) lida com diversos desafios durante o processo de produção de um software. Um deles é a incumbência de definir qual tarefa deverá ser executada por cada membro do time, um problema relevante no gerenciamento de projetos de software. Tal decisão é complexa pois envolve um alto número de variáveis, como os diferentes níveis de habilidades de cada empregado e as diversas características de uma atividade. A Engenharia de Software Baseada em Busca, ou *Search Based Software Engineering* (SBSE), é uma campo que propõe a resolução de problemas complexos da ES como problemas de Otimização Matemática. Sendo assim, este trabalho propõe uma abordagem multiobjetivo que visa minimizar o tempo e o custo de um projeto de software buscando alocar tarefas compatíveis e similares para os empregados. Foi conduzido um estudo empírico para investigar a efetividade da formulação matemática da nova abordagem utilizando os algoritmos NSGA-II, MOCell e IBEA, além de uma Busca Aleatória. Inicialmente foi feito um comparativo da qualidade das soluções encontradas por cada uma das técnicas de otimização. Posteriormente foi investigado a influencia das métricas de qualidade das alocações, Eficiência e Similaridade. Os resultados mostraram que todas as técnicas de busca conseguem encontrar soluções quase ótimas, e que as métricas de qualidade são importantes nesse processo.

Keywords: alocação de recurso humanos. otimização multiobjetivo. similaridade de tarefas.

ABSTRACT

The Software Engineering (SE) addresses a number of challenges during the software production process. One of them is the task of defining which task should be performed by each team member, a relevant problem in the management of software projects. Such a decision is complex because it involves a large number of variables, such as the different skill levels of each employee and the various characteristics of an activity. The *Search Based Software Engineering* (SBSE), is a field that proposes the resolution of SE complex problems as Mathematical Optimization problems. Thus, this work proposes a multiobjective approach that aims to minimize the time and cost of a software project seeking to allocate compatible and similar tasks to employees. An empirical study was conducted to investigate the effectiveness of the mathematical formulation of the new approach using the NSGA-II, MOCell and IBEA algorithms, in addition to a Random Search. Initially a comparison was made of the quality of the solutions found by each of the optimization techniques. Subsequently, the influence of allocation quality metrics, Efficiency and Similarity was investigated. The results showed that all search techniques can find suboptimal solutions, and that quality metrics are important in this process.

Keywords: human resource allocation. multi-objective optimization. tasks similarities.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
cGA	*Cellular Genetic Algorithm*
CPM	*Critical Path Method*
ES	Engenharia de Software
EVA	*Earned Value Analysis*
GD	*Generational Distance*
HV	*Hypervolume*
IBEA	*Indicator-Based Evolutionary Algorithm*
MOCell	*Multi-Objetive Cellular Genetic Algorithm*
NSGA	*Non-dominated Sorting Genetic Algorithm*
NSGA-II	*Non-dominated Sorting Genetic Algorithm-II*
PERT	*Project Evaluation and Review Technique*
RASORP	*Resource Allocation for Software Release Planning*
SBSE	*Search Based Software Engineering*
SP	*Spread*
SQL	*Structured Query Language*

LISTA DE SÍMBOLOS

α	Peso da eficiência sobre a estimativa de tempo da tarefa
β	Peso da similaridade sobre a estimativa de tempo da tarefa

SUMÁRIO

1 INTRODUÇÃO

Segundo Sommerville (2010) a Engenharia de Software (ES) é a disciplina responsável por todos os aspectos ligados a produção de software, a partir de sua especificação, passando pelo seu desenvolvimento, até sua manutenção. A ES não leva em consideração apenas técnicas de desenvolvimento, mas um contexto mais amplo que envolve todo o gerenciamento e desenvolvimento de ferramentas, além de métodos para provê suporte a produção de software. Dentro deste gerenciamento, a gestão de recursos humanos[1] é um componente chave de um projeto, e a falha em projetos desta natureza está frequentemente ligada a um planejamento inadequado de recursos humanos (TSAI; MOSKOWITZ; LEE, 2003).

A metodologia ágil é um dos diversos modelos de desenvolvimentos descritos pela ES, que utilizam abordagens incrementais para desenvolvimento de software, onde em um determinado período de tempo um novo incremento chamado *"release"* é liberado ao produto (FOWLER; HIGHSMITH, 2001). Dado que (i) a produtividade pode variar significantemente entre os empregados (ACUNA; JURISTO; MORENO, 2006) e (ii) podem existir tarefas similares entre si (DAHLSTEDT; PERSSON, 2005), uma boa alocação de recursos necessita contemplar esses dois aspectos e ajudar que o planejamento da *release* seja seguido eficientemente. Tendo em vista todas essas nuances relacionadas a alocação de recursos, pode-se mencionar o Problema da Alocação de Recursos para Planejamento de Release de Software, do inglês *Resource Allocation for Software Release Planning* (RASORP), que consiste em encontrar uma atribuição ótima para os recursos realizarem as tarefas da *release* (NGO-THE; RUHE, 2009).

Diversos problemas da Engenharia de Software, incluindo o RASORP, tem sido recentemente abordados pela subárea chamada *Search Based Software Engineering* (SBSE), ou Engenharia de Software Baseada em Busca, formalizada por Harman e Jones (2001). A SBSE afirma que problemas complexos da ES podem ser reformulados como problemas de busca, e assim serem solucionados com a aplicação de técnicas de otimização matemática, fornecendo soluções de boa qualidade e automatizadas.

1.1 MOTIVAÇÃO

Um dos principais problemas enfrentados pelas empresas em desenvolvimento é definir as tarefas que serão atribuídas a cada empregado (NGO-THE; RUHE, 2009). Se este

[1] Este trabalho considera recursos humanos os diferentes tipos de profissionais envolvidos em projetos de software, como desenvolvedores, analistas, colaboradores externos e etc. Entretanto, como forma de simplificação, será chamado por *empregados*.

problema não for tratado, a eficiência do projeto pode não ser alcançada porque os recursos estão envolvidos com tarefas em que suas capacidades não são maximizadas (KANG; JUNG; BAE, 2011).

Uma equipe de desenvolvimento é geralmente composta por indivíduos com habilidades e níveis diferentes, e cada tarefa também possui exigências e características distintas. Esses fatos dificultam a tarefa de decidir quem ficará responsável por cada atividade. O problema, conhecido como RASORP , tem sido tradado de diferentes formas pela SBSE, por exemplo nos trabalhos de (NGO-THE; RUHE, 2009), (KANG; JUNG; BAE, 2011) e (CHEN; ZHANG, 2013).

Além de considerar as capacidades dos recursos humanos no momento da alocação, outros aspectos podem ser levados em consideração visando o aumento da produtividade da equipe, sejam aspectos referentes a relação recurso humano com a tarefa, ou até mesmo tarefa com tarefa. Um exemplo do segundo relacionamento é a questão da similaridade entre tarefas. Assim como fornecer a um empregado um conjunto de tarefas compatíveis com suas habilidades tende a aumentar a sua produtividade, o fato deste conjunto de atividades serem similares entre si também contribuem para esse fato, visto que os conhecimentos adquiridos empiricamente durante a execução de uma tarefa, podem ser usados em outras.

Contudo, uma das principais limitações das atuais abordagens de SBSE para o RASORP é não explorar as similaridades das tarefas durante a atribuição delas para os recursos humanos. Baseado nestes fatos, este trabalho parte do pressuposto que a alocação de um conjunto de tarefas similares adequadas às habilidades do empregado pode colaborar para a alcançar um planejamento eficiente.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho propõe desenvolver uma abordagem multiobjetivo para resolução do *Resource Allocation for Software Release Planning* (RASORP). Tal abordagem baseia-se na utilização de métodos de otimização matemática para fornecer soluções automatizadas que, além de atribuir tarefas compatíveis com cada empregado, atribui a ele um conjunto de atividade similares, visando minimizar o tempo e custo de cada *release*, e assim do projeto como um todo.

1.2.2 Objetivos Específicos

Para se atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

a) Elaborar uma formulação matemática que seja capaz de atribuir um conjunto de tarefas similares e compatíveis com as habilidades de cada empregado.
b) Implementar três algoritmos evolucionários (NSGA-II, MOCell e IBEA) para a formulação elaborada.
c) Realizar um estudo empírico que evidencie:
    - Qual das técnicas de busca produz melhores resultados.
    - Qual o real impacto de buscar alocações de tarefas eficientes na resolução do *Resource Allocation for Software Release Planning* (RASORP).
    - Qual o real impacto da utilização da similaridade de tarefas na resolução do *Resource Allocation for Software Release Planning* (RASORP).

2 FUNDAMENTAÇÃO TEÓRICA

2.1 OTIMIZAÇÃO MATEMÁTICA

A otimização é o campo do conhecimento que através de suas técnicas e em determinados domínios visa determinar os extremos mínimos ou máximos de funções. Na prática, o extremo que se deseja descobrir de uma função representa algum mérito de um sistema que se deseja construir ou analisar (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012). Deb (2001) refere-se a otimização como a busca por uma ou mais soluções que correspondem aos valores extremos de um ou mais objetivos (funções).

De acordo com Bhatti (2012) formular um problema de otimização envolve os seguintes passos:

(a) Selecionar uma ou mais variáveis de decisão;

(b) Escolher uma função objetivo;

(c) Identificar um conjunto de restrições.

A função objetivo e as restrições precisam sempre estar relacionadas a uma ou mais variáveis de decisão. Assim sendo, uma formulação é comumente descrita da seguinte forma:

$$\begin{aligned} &\text{minimizar (ou maximizar)} && f(x), \\ &\text{sujeito a:} && g_i(x) \leq 0 \text{ para } i = 1,2,3,...,m \\ & && h_j(x) \leq 0 \text{ para } j = 1,2,3,...,p \\ & && x \in \Omega \end{aligned}$$

onde $f(x)$ é a função para a qual se deseja encontrar um valor ótimo, $g(x)$ e $h(x)$ correspondem as restrições do problema e Ω é o conjunto de domínio das variáveis de decisão.

Quando a modelagem de um problema envolve apenas uma função objetivo, a tarefa de encontrar uma solução ótima é chamada otimização mono-objetivo. Por sua vez, quando envolve mais de um objetivo, é chamado otimização multiobjetivo.

2.1.1 Otimização Multiobjetivo

Um problema de otimização multiobjetivo possui um certo número de objetivos que devem minimizados ou maximizados. Erroneamente pode se assumir que o propósito desse tipo de problema é encontrar uma solução ótima correspondente a cada função objetivo, quando

na verdade a finalidade é otimizar todos os objetivos simultaneamente. Em princípio o espaço de busca no contexto multiobjetivo é dividido em duas regiões que não se sobrepõem, uma região ótima e outra não ótima. Por natureza, a presença de múltiplos objetivos dá origem a um conjunto de solução ótimas, chamada Frente de Pareto (DEB; SINDHYA; HAKANEN, 2016).

No caso de existir múltiplas soluções ótimas, é difícil determinar qual delas é melhor. Diferentes soluções podem produzir um *trade-off* entre os distintos objetivos, uma solução que é melhor em um objetivo, pode ser pior em outro. Sendo assim, não havendo nenhuma informação complementar que determine para onde a escolha da melhor solução deve tender, todas as soluções são igualmente importantes (DEB, 2001).

Portanto, visando a abordagem ideal, é importante encontrar o maior número possível de soluções ótimas. Assim, pode se imaginar que uma abordagem multiobjetivo possui dois alvos:

(a) Encontrar um conjunto de soluções o mais próximo possível da Frente de Pareto ótima;
(b) Encontrar um conjunto de soluções o mais diversa possível.

O primeiro alvo é obrigatório em qualquer tarefa de otimização. Convergir para um conjunto de soluções que não é próximo do ideal é indesejável. Somente quando se converge para próximo do ótimo real, que se pode ter certeza de sua quase otimalidade. Esse alvo se assemelha ao objetivo de otimização dos problemas mono-objetivo. Por sua vez, o segundo alvo é particular da otimização multiobjetivo. Além de buscar um conjunto de soluções próxima à frente ótima, elas também devem ser bem distribuídas na região ótima. Apenas se pode garantir um bom *trade-off* entre os objetivos caso a diversidade das soluções sejam garantidas (DEB; SINDHYA; HAKANEN, 2016).

2.1.2 Métricas

Segundo Abraham e Jain (2005) a definição de métricas adequadas é extremamente importante para avaliar um algoritmo. No entanto, quando se trata de otimização multiobjetivo, ela traz consigo alguns problemas para avaliação qualitativa dos resultados. Primeiramente é que será gerada varias solução igualmente boas, e não apenas uma. Segundo, quando se usa técnicas estocásticas, cada execução pode trazer resultados bem diferentes, tornando necessário a execução do processo repetidas vezes. Finalmente, pode-se estar interessado em medir coisas diferentes, como a capacidade do algoritmo em manter diversidade, ou que procure soluções com qualidade equivalentes.

A literatura fornece algumas métricas ja estabelecidas e previamente validadas, por exemplo:

- ***Generational Distance* (GD)**: *Generational Distance* é uma forma de estimar quão longe os elementos do conjunto de soluções não dominadas (Frente de Pareto) está do conjunto definido como sendo ótimo. Um GD = 0 indica que todas as elementos gerados estão no conjunto ótimo, qualquer outro valor define o quão longe estamos do ótimo global de nosso problema (COELLO; VELDHUIZEN; LAMONT, 2002).
- ***Spread* (SP)**: Com esta métrica, se deseja medir a distribuição da Frente de Pareto, desde seu inicio até seu final. A métrica julga quão boa está a distribuição comparando a variância entre cada vizinho do conjunto de soluções não dominadas. Caso essa métrica esteja com valor zerado, indica que todos os membros da frente são equidistante espaçados (ABRAHAM; JAIN, 2005).
- ***Hypervolume* (HV)**: O *Hypervolume* esta relacionado com a área de cobertura da Frente de Pareto no que diz respeito ao objetivo. O calculo é feito criando áreas retangulares de um ponto de referência até cada uma das soluções, o somatório das áreas de todos os retângulos é o resultado do HV (COELLO; VELDHUIZEN; LAMONT, 2002).

2.1.3 Técnicas de Busca

A literatura fornece diversas técnicas para resolução de problemas de otimização, que podem ser divididas em três tipos:

(a) **Enumerativos**: apesar de ser determinístico, é feito uma distinção pois não usam heurísticas.
(b) **Determinísticos**: Algoritmos Gulosos, e Subida de Colina por exemplo.
(c) **Estocástico**: Têmpera Simulado, Algoritmo Genético, entre outros.

Dentre esses, os esquemas enumerativos são as estratégias mais simplórias, mas quem podem encontrar soluções ótimas em um espaço de busca limitado. Entretanto, facilmente se percebe sua ineficiência e até inviabilidade em grandes espaços de busca. Devido a natureza dos problemas do mundo real, muitas vezes é necessário limitar o espaço de busca e admitir encontrar soluções "aceitáveis"em um tempo "aceitável". Os algoritmos determinísticos conseguem se adequar a esse contexto incorporando o conhecimento do domínio do problema. Porém, quando se deparam com problemas complexos, com espaço de busca grande e descontinuo, são prejudicados pela exigência do conhecimento do domínio (DEB, 2001).

Como a maioria dos problemas reais da ciência são irregulares, técnicas de busca enumerativas e determinísticas se tornam inadequadas. Para esses problemas foram desenvolvidos os métodos estocásticos, como a Têmpera Simulada, Busca Tabu e Algoritmos Evolucionários. Estes métodos de forma geral fornecem boas soluções para uma vasta gama de problemas com os quais algoritmos determinísticos foram ineficientes (COELLO; VELDHUIZEN; LAMONT, 2002).

2.2 COMPUTAÇÃO EVOLUCIONÁRIA

Computação evolucionária e uma área de pesquisa inspirada no processo de evolução natural, a metáfora fundamental está relacionada ao poder evolutivo intrínseco de solução de problema, com tentativa e erro. A base da computação evolucionária está na teoria da evolução das espécies de Charles Darwin, em que dado um ambiente que pode hospedar um certo numero de indivíduos, que por instinto se reproduzem, a seleção se torna inevitável para que a população não aumente exponencialmente. Consequentemente, sobreviverão apenas os indivíduos que aproveitem melhor os recursos do ambiente, ou seja, que estejam mais adaptados. Este fenômeno é definido como Seleção Natural (EIBEN; SMITH *et al.*, 2003).

Com base nesses conceitos foram elaboradas técnicas de otimização que evoluam suas soluções, e que o conhecimento adquirido seja transmitido ao longo das gerações, possibilitando assim a evolução das espécies. Alguns exemplos desses algoritmos são o *Non-dominated Sorting Genetic Algorithm-II* (NSGA-II) e o *Indicator-Based Evolutionary Algorithm* (IBEA) (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012).

2.3 ALGORITMO GENÉTICO

Baseado na teoria da evolução das espécies de Charles Darwin, (HOLLAND, 1975) propôs o Algoritmo Genético (AG). Os AG's fazem parte da classe de algoritmos evolucionários, que usam modelos computacionais dos processos naturais de evolução para resolução de problemas. Apesar de existirem vários modelos já propostos, todos utilizam seleção, mutação e reprodução para simular a evolução da espécies (LINDEN, 2008).

De acordo com Gaspar-Cunha, Takahashi e Antunes (2012) os AG's processam o espaço de busca a procura de potenciais soluções para o problema, cada possível solução constitui um indivíduo, e o conjunto de indivíduos são chamados de população. A população é evoluída ao longo de várias interações, simulando gerações, com o objetivo de encontrar uma

solução de qualidade elevada para o problema. Geralmente a população inicial é escolhida de forma aleatória, e evolui de acordo com dois princípios básicos descritos por Darwin, são eles:

(a) Indivíduos mais aptos, que melhor se adaptam ao meio ambiente, têm maior probabilidade se sobreviver. Dentro dos AG's esse principio pode ser simulado aumentando a probabilidade de indivíduos mais aptos serem selecionados para reprodução.

(b) Operadores de transformação, chamados de operadores genéticos, atuam sobre indivíduos da população originando novas soluções. Os já citados mutação e reprodução.

Para que seja possível avaliar a evolução da população, é necessário atentar para dois aspectos:

- **Representação da solução**: como representar as soluções que fazem parte do espaço de busca (Holland defendia a utilização de estruturas binárias).
- **Função de avaliação**: como medir a qualidade de uma solução, representando a capacidade de resolver o problema.

O Algoritmo 1 resume todo o processo do Algoritmo Genético.

Algoritmo 1: Algoritmo Genético

```
Saída: Indivíduo com maior aptidão
início
    Criar população Inicial;
    Calcula aptidão dos indivíduos da população;
    repita
        Faz a seleção dos pais;
        Faz o cruzamento dos indivíduos da população;
        Faz a mutação dos filhos;
        para cada filho inválido faça
            Trata indivíduo inválido;
        fim
        Atualiza população com novos filhos;
        Calcula aptidão dos indivíduos da população;
    até atingir o critério de parada;
    retorna indivíduo com maior aptidão da população
fim
```

Enquanto o mecanismo de seleção, como elitismo, visa aumentar a qualidade média da população durante as gerações, os operadores genéticos são responsáveis por renovar a

população, e garantir sua diversidade. O AG utiliza dois tipos fundamentais de operadores, recombinação e mutação. A recombinação, também conhecida como cruzamento, troca material genético entre dois progenitores, criando assim dois novos indivíduos. Estes indivíduos possuem sequencias genéticas parciais de cada um dos originais. Diferente do cruzamento, a mutação opera em cima de apenas um indivíduo, alterando ligeiramente algumas de suas características. Este é um processo completamente aleatório justamente para garantir a diversidade da população (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012).

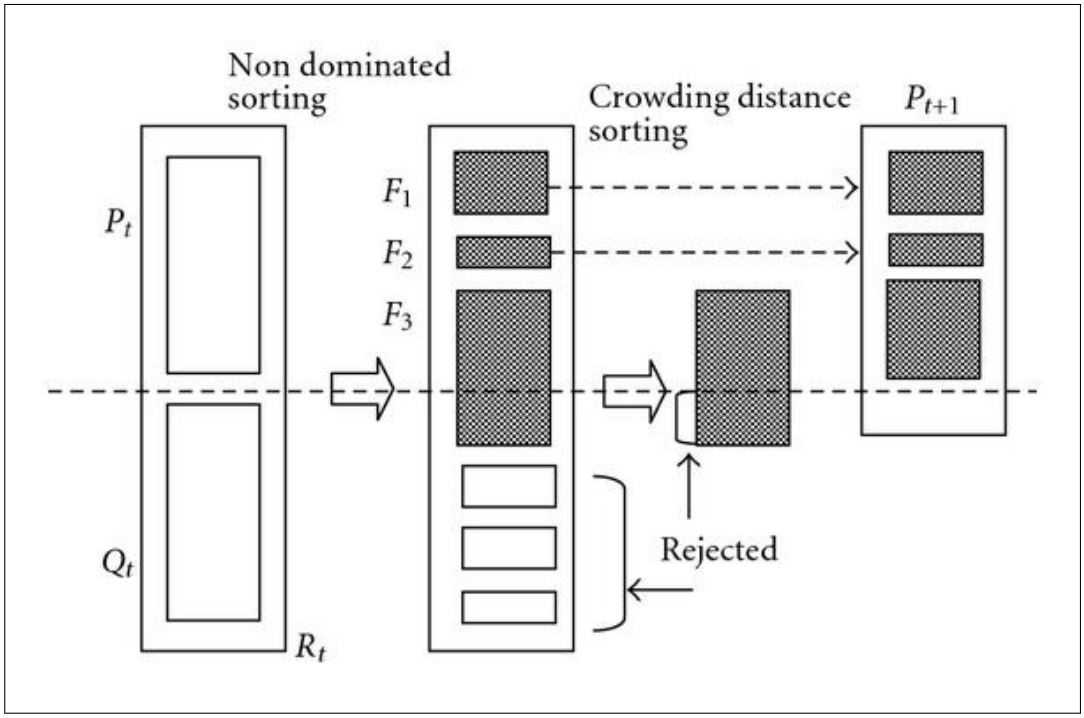
Segundo Gaspar-Cunha, Takahashi e Antunes (2012) um AG tende converge naturalmente para regiões onde existem soluções mais promissoras a medida que as gerações vão passando, parando somente quando atingir o critério de parada estabelecido. Os critérios mais comuns são limite de número de gerações, obtenção de uma solução com a aptidão desejada e a inexistência de melhoria da população por um determinado número de gerações. Quando o critério de parada é atingido, é chegada a hora de apresentar a melhor solução encontrada. A primeira forma de fazê-lo, é mostrando a solução mais apta da última população, ou um conjunto de soluções composto pelos melhores indivíduos.

2.3.1 NSGA-II

O *Non-dominated Sorting Genetic Algorithm-II* (NSGA-II), versão multiobjetivo do algoritmo genético, foi proposto por Deb *et al.* (2000), como uma evolução do *Non-dominated Sorting Genetic Algorithm* (NSGA) proposto por Srinivas e Deb (1994). Esta melhoria foi necessário pois o NSGA ao longo dos anos as seguintes críticas foram repetidamente feitas: (i) alta complexidade computacional, (ii) falta de elitismo e (iii) necessidade de especificar o parâmetro de compartilhamento (DEB *et al.*, 2002).

O NSGA-II solucionou esses problemas utilizando uma abordagem baseada em uma ordenação elitista da população. O algoritmo classifica todas as soluções em diferentes níveis da seguinte forma: no primeiro se encontram todas as soluções não dominadas; no segundo, aquelas que são dominadas por apenas uma solução; e assim sucessivamente, até que todas as soluções estejam classificadas. Esses níveis serão utilizados para a geração das populações posteriores, solucionando assim as críticas relacionadas ao elitismo. Visando manter a diversidade das gerações posteriores e a substituição do parâmetro de compartilhamento, foi proposta uma técnica chamada *crowding distance*, que calcula o espaçamento das soluções dentro de cada nível (DEB *et al.*, 2000).

Figura 1 – Representação NSGA-II



Fonte – Deb *et al.* (2002).

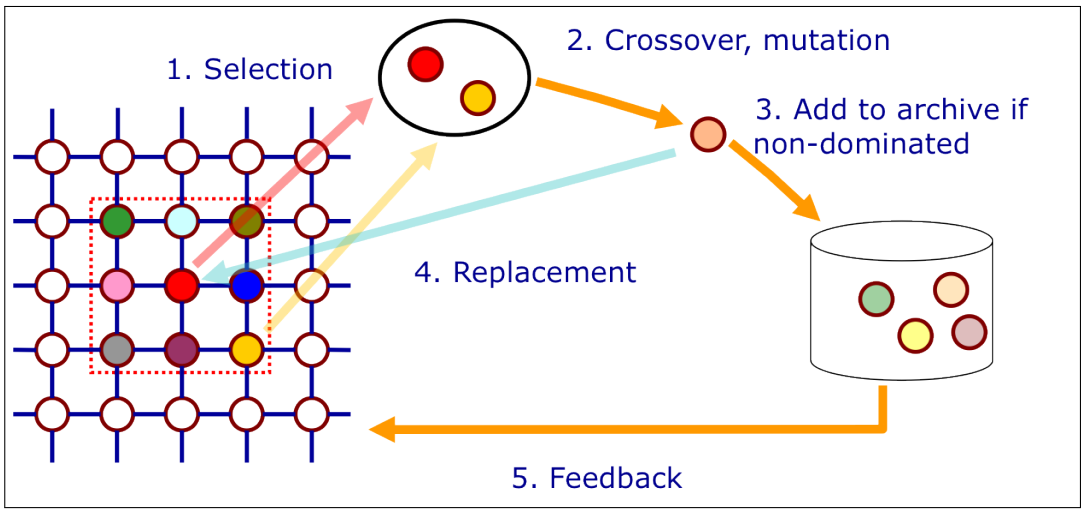
A Figura 1 descreve o funcionamento do NSGA-II. O algoritmo inicia com uma população P_t de tamanho N, a ação dos operadores genéticos sobre ela resulta em uma nova população dita G_t também de tamanho N. A união destas duas populações forma uma terceira chamada R_t de tamanho $2N$. Neste momento é realizado o processo de classificação das soluções como previamente descrito, seguido pela ordenação de acordo com o *crowding distance* dentro de cada nível. A nova população sera formada pelas primeiras N soluções ordenadas. Esse processo é repetido até que o critério de parada seja atingido (DEB *et al.*, 2002).

2.4 MOCELL

Assim como os algoritmos genéticos, os algoritmos celulares também fazem parte da Computação Evolucionaria. O modelo celular simula a evolução natural do ponto de vista do indivíduo. A ideia essencial é prover uma população utilizando uma estrutura conectada, por exemplo um grafo, onde cada indivíduo se comunica com seu vizinho mais próximo (ALBA; DORRONSORO, 2009).

O algoritmo *Multi-Objetive Cellular Genetic Algorithm* (MOCell) foi proposto por Nebro *et al.* (2009), e é uma adaptação multiobjetivo do *Cellular Genetic Algorithm* (cGA). O processo evolucionário pode ser visto no Algoritmo 2. Inicialmente é gerada uma Frente de

Figura 2 – Representação MOCell



Fonte – Nebro *et al.* (2009).

Pareto vazia, e indivíduos são arranjados na malha. Posteriormente, o algoritmo percorre malha passando por cada indivíduo, para cada um deles são selecionados dois vizinhos para gerar um novo filho, em cima dele é efetuado a mutação e analisado sua aptidão. Essa nova solução é adicionada em uma lista auxiliar e em um arquivo externo que guarda todas as soluções não dominadas. Na próxima interação a população é substituída através do mecanismo de *feedback* por indivíduos da lista auxiliar e do arquivo externo. Esse processo, representado na Figura 2, continua até que seja atingido o critério de parada.

2.5 IBEA

O *Indicator-Based Evolutionary Algorithm* (IBEA) é um algoritmo evolucionário multiobjetivo proposto por Zitzler e Künzli (2004). Seu principal diferencial é a utilização de algum indicador arbitrário, além do costumeiro valor de aptidão da solução, durante o processo de otimização. Portanto é necessário que seja definido esse indicador, que pode ser por exemplo o *Hypervolume* ou *Generational Distance*.

De acordo com Thiele *et al.* (2009) o conceito principal do IBEA é formalizar as preferências de acordo com uma relação de domininação. Baseado em um indicador uma aptidão $F(x)$ é calculada para cada indivíduo em relação ao resto da população. O IBEA pode ser descrito da seguinte forma:

Entrada: Tamanho da população a; numero máximo de gerações N.

Saída: Frente de Pareto ótima A.

Algoritmo 2: Algoritmo *Multi-Objetive Cellular Genetic Algorithm*

```
início
    Criar frente de Pareto vazia;
    repita
        para cada indivíduo faça
            Seleciona a lista de vizinhos;
            Faz a seleção dos pais;
            Faz a recombinação dos pais;
            Faz a mutação do filho;
            Insere filho na lista auxiliar;
            Insere filho no arquivo externo;
            Insere na frente de Pareto;
        fim
        Atualiza a população com a lista auxiliar;
        Atualiza a população com o arquivo externo;
    até atingir o critério de parada;
fim
```

Passo 1 - Inicialização: Gerar um população inicial P de tamanho a; Definir contador de gerações $m = 0$.

Passo 2 - Atribuição de aptidão: Calcular *fitness* de todas os indivíduos considerando o indicador escolhido.

Passo 3 Seleção de Ambiente: Realizar os seguintes três passos enquanto a população exceder a:

1. Escolher individuo com menor valor de *fitness*;
2. Remove-lo;
3. Recalcular *fitness* da população;

Passo 4 Finalização: Se $m \geq N$ ou qualquer outro critério de parada for satisfeito, atribuir as soluções não dominadas de P em A. Parar.

Passo 5 Seleção: Executar torneio binário com elementos de P para a criação de P'.

Passo 6 Variação: Aplicar operadores genéticos em P' e adicionar soluções resultantes em P; Incrementar m; Seguir para o Passo 2.

2.6 ENGENHARIA DE SOFTWARE BASEADA EM BUSCA

Engenheiros de software frequentemente enfrentam problemas inerentes a sua área, como balanceamento entre restrições concorrentes ou conflitantes e requisitos imprecisos. Por esse motivo, soluções perfeitas são geralmente impossíveis ou impraticáveis, e como outras disciplinas da engenharia, a ES busca soluções próximas da ótima ou que se enquadrem dentro de uma tolerância aceitável (HARMAN; JONES, 2001).

Os aspectos acima descritos tornam útil a aplicação de técnicas baseadas em buscas, normalmente utilizadas em problemas de otimização matemática, para resolução de problemas da Engenharia de Software. Foi nesse sentido que Harman e Jones (2001) formalizaram a subárea da ES *Search Based Software Engineering* (SBSE), ou Engenharia de Software Baseada em Busca. Além de formalizar, este trabalho demonstra como é conceitualmente simples reformular um problema da ES como um problema de otimização. Uma vez que essa reformulação foi feita, se torna viável a aplicação de meta-heurística em uma grande quantidade de problemas da Engenharia de Software. Porém, alguns elementos são necessários para fazer esta reformulação possível, são eles:

(a) **Representação da solução**: é a definição computacional de uma solução do problema.

(b) **Função de avaliação**: é a forma com que a solução é avaliada, através dele é possível comparar soluções candidatas.

(c) **Operadores**: são utilizados durante o processo evolucionário para gerar uma solução nova a partir de uma já existente, buscando a exploração do espaço de busca.

Segundo Harman e Clark (2004) os trabalhos de SBSE são guiados pelas funções de avaliação. Algumas métricas dessas avaliações são familiares a ES outras porém não são tão habituais, e a cada novo trabalho surgem novas. Diante disso, foi estabelecido uma série de propriedades que devem possuir para que sejam úteis como função de avaliação. São elas:

- **Grande espaço de busca**: Em espaços de busca pequenos não se justifica o uso de uma técnica baseada em busca, já que este pode ser explorado exaustivamente .
- **Baixa complexidade computacional**: Já que para justificar o uso de SBSE é necessário um grande espaço de busca, a complexidade computacional da função de avaliação deve ser baixa, pois ela possivelmente será calculada centenas de milhares de vezes.
- **Continuidade aproximada**: Toda otimização baseada em busca baseia-se na orientação dada pela função de avaliação. Quanto mais descontínua, menor sua contribuição.
- **Ausência de soluções ótimas conhecidas**: Indiscutivelmente, não faz nenhum sentindo

aplicar uma meta-heurística , que não garante uma solução ótima, se já existe um método conhecido que entrega uma solução ótima.

A simplicidade e prática aplicabilidade da SBSE levou a um dramático aumento na investigação desta área. A formalização da área em 2001 não somente intensificou as atividades da comunidade, como também aumentou a diversidade delas. Hoje a SBSE em sido aplicado em diversos ramos da Engenharia de Software, como engenharia de requisitos, planejamento de estimativa e custos, avaliação de qualidade, entre outros (HARMAN, 2007).

2.7 ANÁLISE ESTATÍSTICA DE META-HEURÍSTICAS EM SBSE

Por natureza, as meta-heurísticas possuem um caráter estocástico. Dessa forma, diferentes execuções dessas técnicas tendem a gerar resultados também diferentes. Como são elas as principais ferramentas utilizadas pela SBSE, sempre que se propõe a fazer análise de resultados gerados por diferentes meta-heurísticas, é indispensável que seja feita uma criteriosa comparação dos resultados. Uma forma de garantir que não haja conclusões imprecisas, é com o uso de testes estatísticos.

Para que seja possível realizar um teste estatístico é necessário que os algoritmos que serão comparados sejam executados um certo número de vezes e de forma independente, para que se possa coletar informações sobre a distribuição de probabilidade para cada algoritmo. Após as execuções o teste avalia se existe evidencia empírica suficiente para garantir que há diferença entre os resultados de cada algoritmo, e assim afirmar que um é melhor que o outro. A principio assume-se a hipótese nula H_0 de que não existe diferença entre as amostras, e a função do teste estatístico é verificar se devemos rejeitar a hipótese nula (ARCURI; BRIAND, 2011).

Segundo Arcuri e Briand (2011) existem dois tipos de erros possíveis quando executados testes estatísticos: (i) Rejeita-se a hipótese nula quando ela é verdadeira, e (ii) aceitarmos H_0 quando ela é falsa. Tradicionalmente, há mais ênfase em não cometer o primeiro equívoco. Para isso é definido um *p-value* que indica a probabilidade de um erro deste tipo, e o nível de significância α que representa o maior valor de *p-value* aceito para rejeitar a hipótese nula.

Como um exemplo de teste pareado, pode-se citar o teste de Wilcoxon que analisa se existe uma diferença estatisticamente significativa entre duas amostras. Porém, na maioria das situações pesquisadores precisam comparar vários algoritmos, o que geraria um numero alto de comparações'. Todavia, existem opções para se comparar múltiplas amostras, por exemplo o teste de Kruskal-Wallis (ARCURI; BRIAND, 2014).

Quando compara-se dois algoritmos diferentes e executa-se cada um deles várias vezes, é improvável que tenham a mesma distribuição de probabilidade. Sendo assim, com um bom número de execuções podemos obter diferença estatística mesmo que ela seja na prática irrelevante. Portanto, embora seja extremamente importante avaliarmos se existe diferença estatística entre dois algoritmos, também é crucial analisar a magnitude desta diferença, chamada de *effect size*. O teste de Vargha-Delaney's $\hat{A}_{12}$ é um dos utilizados para mensurar esta propriedade (ARCURI; BRIAND, 2014).

2.8 RESOURCE ALLOCATION FOR SOFTWARE RELEASE PLANNING (RASORP)

O planejamento do projeto constitui uma tarefa vital da engenharia de *software*, desde o desenvolvimento até sua manutenção. A falta de planejamento pode causar atrasos e assim custos inaceitáveis ao projeto. Existem varias ferramentas que auxiliam no planejamento do projeto, como *Project Evaluation and Review Technique* (PERT), *Critical Path Method* (CPM) e *Earned Value Analysis* (EVA). Entretanto, apesar de serem importantes falham em atividades importantes, como criar equipes de trabalho, ou distribuir tarefas entre a equipe (PENTA; HARMAN; ANTONIOL, 2011).

De acordo com Tsai, Moskowitz e Lee (2003) alocação de recursos humanos é um fator chave na redução do custo, duração e risco do projeto. Evidencias apontam que a falha em projetos de software é muitas vezes resultado de um planejamento inadequado da utilização dos recursos humanos, apesar disso, pouca atenção tem sido dada a essa questão. As habilidades técnicas e experiências de cada recurso humano é um fator chave para o sucesso do projeto. Portanto, uma alocação deve ser feita criteriosamente, e considerando os seguintes fatores:

(a) **Escassez de recursos**: recursos humanos em um projeto de software não tem exigido apenas suas habilidades, mas também experiências relacionadas ao projeto. Pessoas mais qualificadas são geralmente um recurso escasso, e maior será a competição por ela entre e dentro de vários projetos.

(b) **Heterogeneidade de recursos**: recursos humanos possuem diferentes pontos fortes e habilidades, educação, experiência, custo e horas disponíveis. O sucesso do projeto e altamente sensível a essas variáveis.

(c) **Não substituição de recursos**: Uma tarefa não é realizada apenas usando as habilidades, o recurso também utiliza-se da aprendizagem contínua ocorrendo dentro da tarefa. Sendo assim, qualquer substituição ou rotação do recurso implica frequentemente em

esquecimento, aumentando a duração do projeto e seu risco.

(d) **Variação de tarefas**: A complexidade em um desenvolvimento de software dificulta a estimativa de tempo para a conclusão de uma tarefa, esse tipo de incerteza deve ser considerada.

É sabido que a produtividade pode variar significantemente entre os recursos, onde cada desenvolvedor pode ter diferente produtividade em diferentes tipos de tarefa. Neste contexto, uma boa alocação assume ainda mais importância. O problema resultante da alocação ótima de recursos para executar uma tarefa dentro de *releases* de software é chamado *Resource Allocation for Software Release Planning* (RASORP), ou Problema da Alocação de Recursos para Planejamento de Release de Software (NGO-THE; RUHE, 2009).

Segundo Kang, Jung e Bae (2011) a alocação de recursos humanos difere e é bem mais complexa que outros tipos de alocação, como planejamento de produção ou distribuição de carga, pois envolve as características humanas. E essas características também devem ser consideradas, e podem ser divididas em:

- **Características individuais**:
  (a) **Curva de aprendizado**: Quando um recurso recebe uma atividade, ele cresce junto com ela. Portanto, será mais eficiente alocar tarefas com a qual ele esteja familiarizado.
  (b) **Níveis diferentes de produtividade**: Recursos possuem produtividades diferentes, e que variam de acordo com o tipo de tarefa que recebem.
- **Características coletivas**:
  (a) **Necessidade de coesão da equipe**: Em desenvolvimento o time deve agir como um só, aumentando a eficiência do desenvolvimento. Portanto, o ideal é que os recursos trabalhem perto e ajudando uns aos outros.
  (b) **Sobrecarga de comunicação**: Excesso de comunicação reduz a eficiência do projeto. O impacto é ainda maior quando esse excesso ocorre entre times diferentes.
  (c) **Necessidade de colaboração e gestão**: A colaboração é necessária, e o desempenho de um recurso pode ser afetado por outros. Assim, é importante alocar desenvolvedores capazes de liderar e auxiliar outros recursos.

O propósito da resolução do RASORP visa aumentar a produtividade do projeto, consequentemente é comum que os objetivos do trabalho sejam diminuir custo e tempo como explorado em (CHEN; ZHANG, 2013), (BIBI; AHSAN; ANWAR, 2014), (ANTONIOL; PENTA; HARMAN, 2004b), entre outros. Esses objetivos geralmente são alcançados atribuindo atividade-

des compatíveis com cada recurso humano.

2.9 CONCLUSÕES DO CAPÍTULO

Neste capítulo foram apresentados os fundamentos do presente trabalho. Inicialmente foi explanado sobre o campo da otimização matemática. Dentro deste campo foi descrito os problemas abordados por ela, com ênfase na otimização multiobjetivo, e apresentado métricas e técnicas utilizadas para solução de problemas.

Posteriormente discutiu-se sobre os três algoritmos utilizados no trabalho, NSGA-II, MOCell e IBEA. Seguido pela apresentação dos conceitos da área *Search Based Software Engineering* (SBSE), que trata problemas da Engenharia de Software (ES) com otimização matemática.

Em seguida, foi discutido sobre a aplicação de análise estatística quando usada meta-heurísticas em SBSE. Finalmente foi descrito o *Resource Allocation for Software Release Planning* (RASORP), complexa questão da Engenharia de Software (ES), cuja abordagem proposta neste trabalha busca solver.

3 TRABALHOS RELACIONADOS

3.1 ALOCAÇÃO DE RECURSOS HUMANOS BASEADA EM RESTRIÇÕES

Uma Têmpera Simulada acelerada baseada em restrições de níveis individuais e coletivos foi avaliada por Kang, Jung e Bae (2011). A abordagem tem como princípio definir varias restrições que devem ser consideradas e penalizadas durante a alocação. O trabalho tem dois tipos de estudo de caso, o primeiro visa mostrar a importância das restrições atribuindo penalidade zero a todas as restrições. Neste caso, quanto maior o numero de penalidades quebradas, maior a importância das restrições. O segundo estudo de caso visa validar a efetividade da abordagem na alocação de recursos, comparando com a duração de projetos reais.

O principal foco do trabalho foi mostrar a importância da utilização de restrições na otimização do problema da alocação de recursos humanos. Este problema naturalmente envolve muitas questões que devem ser levadas em consideração, pois envolve duas variáveis voláteis: (i) requisitos e (ii) pessoas. E como demonstrado neste trabalho, ignorar essas questões acarreta resultados inconsistentes.

3.2 ALOCAÇÃO DE RECURSOS HUMANOS EM MANUTENÇÃO MASSIVA DE SOFTWARE

Otimizar o problema da alocação de recursos humanos com técnicas baseadas em busca não é um novidade em SBSE, nos primórdios da referida área Antoniol, Penta e Harman (2004b) já propuseram uma abordagem sob a perspectiva da manutenção massiva de projetos de software. O trabalho visa alocar um time de programadores a um série de pacotes de trabalho. O estudo de caso foi realizado utilizando um histórico real de atividades para remediar o problema de datas conhecido como Y2K, e foi comparado a performance de três algoritmos baseados em busca: Algoritmo Genético, Têmpera Simulada e Subida de Colina.

Nos anos seguintes os autores deram continuidade a pesquisa com novas publicações. Em Antoniol, Penta e Harman (2004a) propuseram uma técnica robusta dentro do mesmo contexto. Porém, desta vez com objetivo de minimizar o tempo do projeto, além da alocação, a abordagem lidava com incertezas como, abandono de programadores, retrabalho e erros ou incertezas nas estimativas feitas. O estudo empírico foi realizado utilizando a mesma instância do trabalho anterior, porém desta vez apenas um algoritmo genético foi utilizado. Já em (ANTONIOL; PENTA; HARMAN, 2005) também foi utilizado o mesmo contexto, porém

utilizando três técnicas de busca, Algoritmo Genético, Têmpera Simulada e Subida de Colina. Cada uma com dois tipos de representação, a primeira utilizando o principio da casa dos pombos, a segunda com representação por ordenação.

No trabalho de Penta, Harman e Antoniol (2011) foi utilizado técnicas de busca para determinar as pessoas necessárias para terminar o projeto dentro do prazo, dividi-los em equipe e distribuir as tarefas. O objetivo da abordagem foi diminuir o tempo de conclusão e a fragmentação do cronograma. A abordagem foi avaliada utilizando um Algoritmo Genético e uma Têmpera Simulada, em duas instancias reais.

A principal diferença dos trabalhos apresentados até aqui com o presente estudo, está no ambiente do problema. Apesar de buscarem otimizar a alocação dos recursos visando diminuir o tempo de conclusão das atividades, o contexto da manutenção massiva traz variáveis diferentes da encontrada na engenharia de requisitos.

3.3 ALOCAÇÃO DE RECURSOS HUMANOS EM ENGENHARIA DE REQUISITOS

Em Ngo-The e Ruhe (2009) utilizou-se uma combinação de programação inteira linear e genética para otimizar a alocação de recursos junto com planejamento de *release*. A abordagem é dividida em duas fases, na primeira é feita a seleção das atividade que faram parte da *release*, posteriormente, na fase dois, é executada a alocação dos recursos. Na fase 1 a abordagem tem como objetivo selecionar os requisitos que tenham maior valor de negocio para organização, levando em conta a urgência da tarefa, importância para os clientes e oportunidade de mercado estimada. Na segunda a alocação é otimizada visando o aumento da produtividade e diminuição do tempo. Comparando ao presente trabalho a abordagem é capaz de realizar a priorização das tarefas, porém na fase de alocação considera apenas a produtividade dos recursos humanos. Não leva em consideração o custo de cada um deles, nem como isso impacta no projeto.

Em seu trabalho Chen e Zhang (2013) foi desenvolveu um ACO com um esquema de representação baseado em evento que visa resolver tanto o problema da alocação de recursos, como o do planejamento de *release*. O objetivo do trabalho foi a diminuição do custo do projeto, realizando experimentos com instancias reais e artificiais. Assim como o trabalho previamente apresentado também ataca o problema do planejamento de *release*, mas ignora a variável custo no processo de alocação.

Bibi, Ahsan e Anwar (2014) propôs uma abordagem multiobjetivo para resolução

do problema da alocação de recursos, seus objetivos foram: (i) aumentar a utilização dos recursos, (ii) diminuir o tempo do projeto e (iii) diminuir o custo do projeto. Posteriormente, em (BIBI; ANWAR; AHSAN,), como continuação da pesquisa foram comparados três algoritmos chamados, MOGA (versão multiobjetivo do algoritmo genético), MOPSO (versão multiobjetivo da otimização por enxame de partículas) e ENSES. Os três algoritmos foram implementados e utilizados utilizando a ferramenta MATLAB. Os resultados foram primeiramente comparados com alocações feitas com a ferramenta MS Project, e obtiveram melhores resultados. Na comparação entre as meta-heurísticas, o MOPSO atingiu resultado superiores aos demais.

De todos os trabalhos apresentados este ultimo é o que mais se assemelha com esse estudo, pois possui praticamente os mesmos objetivos. Porém, acreditasse que pelo fato deste utilizar o tópico da similaridade e técnicas de buscas diferentes, os resultados encontrados serão superiores.

3.4 CONCLUSÕES DO CAPÍTULO

Neste capítulo foram discutidos trabalhos que de alguma maneira se assemelham ao presente estudo. Inicialmente foi apresentado um trabalho que demonstra a importância de dá efetiva atenção às restrições do problema abordado. Posteriormente discutiu-se sobre abordagens de alocação de recursos humanos no contexto da manutenção massiva de software e suas particularidades.

Finalmente, foram apresentado trabalhos mais próximos deste, que de fato fazem parte da Engenharia de Requisitos. Dois deles que além da alocação realizam planejamento de *release*, e um terceiro que otimiza buscando minimizar tempo e custo.

4 ABORDAGEM PROPOSTA

Como previamente mencionado, métodos ágeis são caracterizados pela entrega de *releases* de uma forma incremental e iterativa. Este trabalho é baseado no método Scrum, no qual o processo de desenvolvimento segue uma série de ciclos chamados *sprints*. Cada *sprint* é uma unidade de planejamento onde o trabalho a ser feito é analisado, atividades são selecionadas, recursos são alocados e uma parte do software é implementado. Ao final de uma *sprint*, as funcionalidades completadas são entregues aos *stakeholders* (SCHWABER, 2004).

Considere $T_h = \{t_1, t_2, t_3, \ldots, t_N\}$ um conjunto de tarefas que devem ser realizadas em uma *sprint* h e $E_h = \{e_1, e_2, e_3, \ldots, e_M\}$ o conjunto de empregados que representam os recursos humanos do projeto, onde N e M são o numero total de tarefas e empregados, respectivamente. Como representação da solução de alocação, considere um vetor $S = \{x_1, x_2, x_3, \cdots, x_N\}$ com $x_i \in \{1, 2, 3, \ldots, M\}$, onde $x_i = m$ indica que a tarefa t_i está atribuída ao empregado e_m.

Cada tarefa t_n possui um numero de horas de desenvolvimento estimada representada pelo vetor $TEH_t = \{teh_1, teh_2, teh_3, \ldots teh_N\}$, e um conjunto de habilidades requeridas descrita na matriz $TSM_{N \times Z}$, onde tsm_{nz} indica o valor da habilidade z exigida pela tarefa t_n. Por sua vez, cada empregado e_m é composto por:

(a) Valor da hora, $EHV_e \in [1, L]$, onde L é o valor máximo pago por hora;

(b) Valor da hora extra, $EXV_e \in [L, K]$, onde K é o valor máximo pago por hora extra;

(c) Carga horária padrão, $ESW_e \in [1, H]$, onde H é o total de horas padrão;

(d) Máximo de horas extras, $EMO_e \in [0, G]$, onde G é o numero máximo de horas extras permitidas;

(e) Matriz de Habilidades $ESM_{M \times Z}$, onde esm_{mz} indica o nível de habilidade z definida para o empregado m.

A definição desses elementos foi baseada em outros trabalhos que lidam com o RASORP, como (NGO-THE; RUHE, 2009) e (KANG; JUNG; BAE, 2011), sendo necessário adaptá-las para considerar o contexto de desenvolvimento ágil, como a definição da carga horária de cada empregado dentro da *sprint*. Na prática, o trabalho assume que os valores de ESM, TSM e TEH são estimados e fornecidos pelo engenheiro de requisitos, e valores financeiros são providos pela empresa.

Adicionalmente, como suporte a abordagem, foram definidas outras quatro estruturas: a primeira é uma matriz de similaridade SM_{NN}, onde N é o numero de tarefas, que guarda o nível de similaridade entre as tarefas. As outras três estruturas são os vetores $ehvVec = \{q_1, q_2, \ldots, q_M\}$,

$exvVec = \{k_1, k_2, \ldots, k_M\}$ e $eswVec = \{y_1, y_2, \ldots, y_M\}$, que armazenam respectivamente os valores de EHV, EXV e ESW para todos os empregados.

Como previamente estabelecido, essa abordagem busca alocar tarefas compatíveis e similares para os empregados. Portanto, a alocação de uma tarefa deve ser feita considerando as habilidades exigidas por ela e possuídas pelo empregado. Se o empregado apresenta um nível em dada habilidade superior ao que é exigido pela tarefa, é esperado que ele/ela tenha menos dificuldade para executá-la, caso contrário a dificuldade tende a ser maior. Pode-se perceber, que este fato afeta diretamente a estimativa de tempo necessária para o desenvolvimento da atividade. Dado uma tarefa t_n e um empregado e_m, a efetividade de uma alocação é definida pela função $EfcFactor(i, x_i)$, do seguinte modo:

$$EfcFactor(i, x_i) = \frac{\sum_{j=1}^{Z} ESM_{x_i,j} - TSM_{i,j} \times requiredSkill(TSM_{i,j})}{\sum_{j=1}^{Z} requiredSkill(TSM_{i,j})}$$

$$requiredSkill(TSM_{i,j}) = \begin{cases} 1, & se\ TSM_{i,j} > 0 \\ 0 & \text{caso contrário.} \end{cases}$$

onde $EfcFactor(i, x_i)$ retorna o resultado do somatório da média da diferença entre o valor de habilidade exigida é a possuída pelo empregado, pelo numero de habilidades. A função $requiredSkill(TSM_{i,j})$ determina se uma habilidade em particular é requerida para a finalização da tarefa, checando se $TSM_{i,j} > 0$.

Outro aspecto que influencia o tempo estimado para cada tarefa é a similaridade entre elas, uma vez que tarefas similares são alocadas juntas para um mesmo empregado, elas tendem a serem desenvolvidas de forma mais eficiente. Baseando-se em uma categoria de interdependência chamada *Similar_to*, vindo das interdependências estruturais de requisitos, que descreve situações onde "um requisito é similar ou sobreposto a outro em termos de como é expresso ou em termos de uma ideia semelhante do que o sistema deve fazer"(DAHLSTEDT; PERSSON, 2005). A função $SimFactor(i, S)$ expressa um fator de similaridade entre todas as tarefas atribuídas juntas a uma determinada tarefa i.

$$SimFactor(i, S) = \begin{cases} \dfrac{\sum_{j=1|j\neq i}^{N} SM_{i,j} \times isCoupling(j, i, S)}{couplingNumber(i, S)}, & se\ couplingNumber(i, S) > 0 \\ 0 & \text{caso contrário.} \end{cases}$$

$$couplingNumber(i,S) = \sum_{j=1|j\neq i}^{N} \times isCoupling(j,i,S)$$

$$isCoupling(j,i,S) = \begin{cases} 1, & x_j = x_i \\ 0 & \text{caso contrário.} \end{cases}$$

onde a função $couplingNumber(i,S)$ determina se existem tarefas alocadas junto com a tarefa i através do retorna da função $isCoupling(j,i,S)$. Se houver, a função $SimFactor(i,S)$ retorna a média do somatório das similaridades entre as tarefas, caso contrário o valor de $SimFactor(i,S)$ é 0.

Para determinar o tempo total consumido pelas tarefas em uma possível solução, é proposto a função $Time(S)$, baseado no somatório de todos os valores de TEH das tarefas alocadas em uma solução S, considerando o impacto imposto por $EfcFactor(i,x_i)$ e $SimFactor(i,S)$. Ambos os fatores se comportam de forma similar. Se seus retornos forem igual a 0, eles não irão afetar o valor de TEH. Se o retorno for maior do que 0, o tempo estimado será reduzido. Entretanto, se o resultado for menor do que 0, apenas a função $EfcFactor(i,x_i)$ influenciará o valor de TEH e, consequentemente, o tempo estimado será estendido. A função $Time(S)$ é da por:

$$Time(S) = \sum_{i=1}^{N} TEH_i \times (1 - \alpha \times EfcFactor(i,x_i)) \times (1 - \beta \times SimFactor(i,S))$$

onde α e β representa os pesos que as funções $EfcFactor(i,x_i)$ e $SimFactor(i,S)$ sobre o valor TEH_i, respectivamente.

Diferente de $Time(S)$, onde o valor de TEH de cada uma das tarefas em uma solução S são consideradas, a função $EstimatedHours(e,S)$ é usada para calcular o TEH das tarefas atribuídas para cada um dos empregados e_m. A função $EstimatedHours(e,S)$ é calculada somando o total de horas estimadas de cada tarefa alocadas, considerando o impacto causado por $EfcFactor(i,x_i)$ e $SimFactor(i,S)$, da mesma forma que em $Time(S)$.

$$EstimatedHours(e,S) =$$
$$\sum_{i=1}^{N} TEH_i \times (1 - \alpha \times EfcFactor(i,e)) \times (1 - \beta \times SimFactor(i,S)) \times isAllocated(e,S)$$

$$isAllocated(e,S) = \begin{cases} 1, & se\ e = x_i \\ 0 & \text{caso contrário.} \end{cases}$$

onde $isAllocated(e,S)$ determina se uma tarefa t_i está alocada a um empregado e.

Dado o numero total de horas trabalhadas por um empregado obtido pela função $EstimatedHours(e,S)$, é possível contabilizar o custo dele/dela para o projeto através da função $ValueHours(e,wH)$:

$$ValueHours(e,wH) = \begin{cases} ehvVec_e \times wH, & se\ wH \leq eswVec_e \\ (exvVec_e \times (wH - eswVec_e)) + & \\ (ehvVec_e \times eswVec_e) & \text{caso contrário.} \end{cases}$$

onde, se a estimativa é que o empregado e irá trabalhar apenas sua carga horária padrão representada por ESW, o total de horas é multiplicado por seu valor de hora EHV. Porém, se ele/ela terá de trabalhar além de sua carga horária, a quantidade extra é multiplicada pelo valor valor de hora extra EXV.

Ademais, propõe-se a função $Cost(S)$ com objetivo de obter o custo total relativo a todos os funcionários que receberam atividades em uma solução S:

$$Cost(S) = \sum_{i=1}^{M} ValueHours(x_i, EstimatedHours(x_i,S)).$$

Finalmente, a formulação multiobjetivo deste trabalho consiste em minimizar o $Time(S)$ e $Cost(S)$, respeitando duas restrições. Este modelo pode ser formalizada da seguinte forma:

$$\begin{aligned} &\text{minimizar} && Time(S), \\ &\text{minimizar} && Cost(S), \\ &\text{sujeito a:} && 1)\ estimatedHours(x_i,S) - eswVec_{x_i} \leq EMO_{x_i} \forall x_i \in S, \\ & && 2)\ ESM_{x_i,j} > 0 \ \forall\ TSM_{i,j} > 0 \end{aligned}$$

onde a primeira restrição garante que o total de hora extra trabalhada por um empregado não seja superior ao permitido, representado por EMO. A segunda restrição previne que uma tarefa

que exige uma habilidade específica seja alocada a um empregado que não tenha essa habilidade em particular.

5 ESTUDO EMPÍRICO

Este capítulo tem como finalidade apresentar detalhes do estudo empírico realizado a fim de legitimar a abordagem multiobjetivo proposta. Inicialmente será apresentado as questões de pesquisa que dirigiram o estudo. Posteriormente será apontado as definições dos experimentos, como instâncias utilizadas, e descrição das configurações aplicadas aos algoritmos de busca.

Na Seção 5.3 serão apresentados os resultados obtidos e suas respectivas considerações, e finalmente a Seção 5.4 discute as conclusões dos experimentos.

5.1 QUESTÕES DE PESQUISA

Para que a abordagem proposta possa ser avaliada os experimentos foram realizados norteados pelas seguintes questões de pesquisa:

QP1: De acordo com a métricas de avaliação definidas, qual dos algoritmos possui melhor desempenho?

QP2: Qual a efetiva influência do fator α nas soluções encontradas?

QP3: Qual a efetiva influência do fator β nas soluções encontradas?

Visando responder estas perguntas, o estudo empírico foi realizado utilizando instâncias reais e artificiais, três algoritmos de busca, e uma busca randômica para fins de validação. As repostas e discussões para cada uma das instâncias se encontram na Seção 5.3.

5.2 DEFINIÇÕES DOS EXPERIMENTOS

5.2.1 Instâncias

O presente estudo empírico foi realizado utilizando quatro instâncias, sendo duas delas reais e duas artificiais. As duas reais foram fornecidas pela empresa Invista Tech[1], e todas as informações necessárias foram dadas e/ou estimadas durante a fase de levantamento de requisitos pelo responsável da organização. As instâncias artificiais foram utilizadas a fim de diversificar os análises dos experimentos, além de validar a efetividade da abordagem com uma quantidade expressiva de atividades e recursos humanos. Tais valores foram determinados de

[1] http://invistatech.com.br/

forma aleatória dentro de um intervalo pré-definido. As quatro instâncias estão disponíveis na pagina de suporte do trabalho[2].

As instâncias reais foram chamadas de *dataset-1* e *dataset-2*, e fazem parte de dois diferentes projetos de software. Os requisitos presentes na primeira instância referem-se a um projeto novo, que será responsável pelo controle de uma comunidade católica. Ao contrário da primeira, a segunda instância diz respeito a um sistema *mobile* de pré-venda que já se encontra em produção, portanto dizem respeito a novos incrementos ao software. Por sua vez as instancias artificiais foram chamadas de *dataset-3* e *dataset-4*, e como previamente dito, tiveram seus valores definidos aleatoriamente.

O *dataset-1* possui 25 requisitos que englobam quatro áreas de conhecimento: (i) *interface*, (ii) regras de negócio, (iii) modelagem de banco de dados e (iv) teste, e três empregados. O *dataset-2* é composto por 32 tarefas, que diferente do *dataset-1* foram divididas nas quatro áreas de conhecimento supracitadas. A equipe responsável pela entrega do projeto é formada por 5 empregados, sendo três desenvolvedores/analista de sistemas e dois analistas de teste. O *dataset-3* possui 50 atividades que devem ser alocadas para 10 empregados, enquanto o *dataset-4* é composto de 100 tarefas e 20 empregados. Estes valores podem ser visto no Quadro 1.

Quadro 1 – Informações sobre as instâncias utilizadas no experimento

	Quantidade de Tarefas	Quantidade de Empregados
Dataset-1	25	3
Dataset-2	32	5
Dataset-3	50	10
Dataset-4	100	20

Fonte – Elaborado pelo autor

Em todas as instâncias cada tarefa e empregado respectivamente requer e possui quatro tipos de habilidades, são elas: (i) *interface*, (ii) lógica, (iii) *Structured Query Language* (SQL) e (iv) teste. Além disso, cada instância possui uma matriz de similaridade entre os requisitos, que diz o quão similar são as tarefas entre si, e informações como estimativa de tempo de cada atividade, valor de hora e carga-horária de cada empregado, entre outros.

5.2.2 Algoritmos de Busca

O estudo foi realizado utilizando as técnicas de busca NSGA-II, MOCell e IBEA, com o auxílio do *framework jMetal* proposto por Durillo e Nebro (2011). A finalidade da

[2] http://goes.uece.br/lucasroque/rabs/en

utilização de tais algoritmos é realizar um comparativo entre eles, e medir qual possui melhor desempenho para a abordagem. Uma descrição do funcionamento destes foi realizada na seção 2 deste trabalho. Como estabelecido por Harman *et al.* (2012) além de avaliar diferentes algoritmos baseados em busca, é interessante utilizar uma Busca Aleatória a fim de validar que os resultados obtidos pelas meta-heurísticas não foram fruto do acaso, funcionando como um teste de sanidade.

Os parâmetros dos três algoritmos foram obtidos de forma empírica e configurados com 256 indivíduos e definido um total de 102400 avaliações como critério de parada. Foi aplicada um cruzamento *one point* com probabilidade de 90% e uma mutação de 1%, onde um gene do indivíduo é aleatoriamente substituído por um valor inteiro dentro das variáveis possíveis.

Para lidar com a natureza estocástica das meta-heurísticas, cada algoritmo foi executado 30 vezes. Primeiramente com α e β fixos em 0.5, foi com as soluções encontradas nessa configuração foi feita a comparação dos algoritmos, visando a resolução da **QP1**. Posteriormente para solucionar a **QP2**, o β foi fixado em 0.5 enquanto α variou entre 0 e 0.9. Finalmente o α foi da mesma forma fixado e o β variado, para resolução da **QP3**.

5.2.3 Métricas e Análise Estatísticas

Como métricas de avaliação dos resultados dos métodos experimentais, foram usadas o *Hypervolume* (HV), *Generational Distance* (GD) e *Spread* (SP), descritas na seção 2 deste trabalho. A frente utilizada como referência para o cálculo das métricas de desempenho das técnicas adotadas foi gerada como proposto por Zhang (2010). Esta frente é gerada unindo os resultados das execuções das três técnicas de busca escolhidas, quando removido todas as soluções dominadas o resultado e a Frente de Pareto real que será utilizada.

A respeito das análises estatísticas, primeiramente foi realizado o teste de Kruskal-Wallis para averiguar a existência de diferença estatística entre as amostras das meta-heurísticas, além da busca aleatória. Após essa comprovação, foi efetuado uma comparação par a par, usando o teste Wilcoxon com o método de ajuste Bonferroni considerando o nível de confiança em 95%. Adicionalmente, foi usado o teste Vargha-Delaney's $\hat{A}_{12}$ para mensurar a significância estatística.

5.3 RESULTADOS E ANÁLISES

Nesta seção serão apresentados os resultados obtidos e suas respectivas análises. Inicialmente serão discutidos os desempenhos das técnicas de busca para cada uma das ins-

tâncias, correspondendo a **QP1**. Em seguida será discorrido sobre as questões **QP2** e **QP3**. Neste momento, por limitação de espaço, não será discutido sobre cada uma das configurações utilizadas, e sim um panorama geral da influência dos fatores de eficiência e similaridade sobre os objetivos. Os resultados de todas as execuções se encontram na página de suporte.

5.3.1 QP1 - De acordo com a métricas de avaliação definidas, qual dos algoritmos possui melhor desempenho?

As tabelas presentes nesta seção mostra a comparações dos algoritmos das linhas com os das colunas para cada uma das métricas definidas, mostrando os resultados estatísticos dos teste de Wilcoxon (WC) e Vargha-Delaney's $\hat{A}_{12}$. Em termos práticos se o valor para o teste de Wilcoxon for menor que 0.05, isso quer dizer que existe diferença estatística entre as amostras. Com relação ao teste Vargha-Delaney's, um resultado de 0.90, indica que 90% escolhendo uma solução de cada algoritmo, a amostra do algoritmo da coluna foi superior. Lembrando que o *Hypervolume* (HV) mede a diversidade e a proximidade da frente gerada pelo algoritmo com a real, quanto maior o HV melhor a qualidade da frente. Enquanto *Spread* (SP) e *Generational Distance* (GD) medem apenas a diversidade e a proximidade com a frente real respectivamente, neste caso quanto menor o seu valor, melhor as soluções.

Na primeira comparação par a par do *dataset-1* verifica-se um desempenho bem próximo entre o MOCell e o IBEA. O primeiro foi superior em GD, enquanto o segundo é melhor no quesito SP. Quando olhamos para métrica HV, percebe-se que não existe diferença estatística entre eles. Pondo a prova o NSGA-II contra o IBEA, observa-se melhor desempenho apenas em *Generational Distance* (GD), porém em relação ao MOCell é superior em 100% dos confrontos em todas as métricas, vide Tabela 2.

Ainda analisando o primeiro *dataset*, verifica-se que a busca randômica consegue melhores resultados em HV e GD frente ao IBEA, e HV e SP comparado ao MOCell.

As análises do *dataset-2* encontram-se disponíveis na Tabela 3, e pode-se ver novamente um desempenho próximo entre as técnicas MOCell e IBEA, porém desta vez o primeiro foi superior em SP e o segundo em GD, na métrica HV novamente a ausência de diferença estatística entre as soluções.

Nesta instância o NSGA-II tem melhor desempenho em HV que seus concorrentes, porém foi inferior à ambos na métrica SP. Em GD foi superior ao MOCell, e pior que o IBEA. Analisando a técnica randômica percebe-se que ela obteve melhores resultados em SP que as

Tabela 2 – Resultados **Dataset-1** dos testes Wilcoxon (WC) e Vargha-Delaney's $\hat{A}_{12}$, comparando o algoritmo linha a coluna para cada métrica de qualidade.

Algoritmos	Métricas	**Dataset-1**					
		IBEA		MOCell		NSGA-II	
		WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
MOCell	HV	1.00E+00	**0.48**	-	-	-	-
	SP	1.70E-10	0.97	-	-	-	-
	GD	2.20E-09	0.14	-	-	-	-
NSGA-II	HV	5.00E-03	0.00	5.30E-03	**0.75**	-	-
	SP	1.10E-05	1.00	1.60E-10	**0.04**	-	-
	GD	1.00E+00	**0.00**	3.70E-09	**0.26**	-	-
Randômico	HV	1.80E-10	**0.75**	1.70E-10	**1.00**	1.70E-10	0.00
	SP	1.80E-10	0.54	9.00E-03	**0.00**	1.70E-10	0.00
	GD	1.80E-10	**0.00**	1.70E-10	1.00	1.70E-10	1.00

Fonte – Elaborado pelo autor

Tabela 3 – Resultados **Dataset-2** dos testes Wilcoxon (WC) e Vargha-Delaney's $\hat{A}_{12}$, comparando o algoritmo linha a coluna para cada métrica de qualidade.

Algoritmos	Métricas	**Dataset-2**					
		IBEA		MOCell		NSGA-II	
		WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
MOCell	HV	1.00E+00	**0.51**	-	-	-	-
	SP	1.30E-10	0.00	-	-	-	-
	GD	1.70E-10	1.00	-	-	-	-
NSGA-II	HV	5.50E-04	**0.79**	1.33E-03	**0.77**	-	-
	SP	5.30E-04	0.79	1.20E-10	1.00	-	-
	GD	4.20E-02	0.70	1.40E-10	**0.00**	-	-
Randômico	HV	1.20E-10	0.00	1.50E-10	0.00	1.50E-10	0.00
	SP	1.60E-10	**0.00**	2.60E-07	**0.09**	1.50E-10	**0.00**
	GD	1.80E-10	1.00	1.70E-10	1.00	1.50E-10	1.00

Fonte – Elaborado pelo autor

técnicas de busca, isso acontece porque o *Spread* mede a diversidade da amostra, favorecendo-a.

Os próximos dois *datasets* dizem respeito às instâncias artificias, os testes estatísticos do primeiro deles chamado *dataset-3* estão na Tabela 4. O resultado da comparação das saídas dos algoritmos MOCell e IBEA segue o padrão das ultimas duas instâncias, assim como a superioridade da busca randômica no quesito SP. A principal diferença que pode-se constatar está na queda de desempenho do NSGA-II que foi inferior ao IBEA em todas as métricas, e foi superior ao MOCell apenas em GD.

No *dataset-4*, Tabela 5, segue a tendência de crescimento do IBEA e queda do

Tabela 4 – Resultados **Dataset-3** dos testes Wilcoxon (WC) e Vargha-Delaney's $\hat{A}_{12}$, comparando o algoritmo linha a coluna para cada métrica de qualidade.

Algoritmos	Métricas	**Dataset-3**					
		IBEA		MOCell		NSGA-II	
		WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
MOCell	HV	1.00E+00	0.52	-	-	-	-
	SP	1.80E-10	0.00	-	-	-	-
	GD	5.30E-09	0.96	-	-	-	-
NSGA-II	HV	1.10E-03	**0.22**	8.00E-05	**0.17**	-	-
	SP	1.20E-01	**0.67**	1.80E-10	**1.00**	-	-
	GD	1.40E-03	**0.78**	9.00E-06	0.14	-	-
Randômico	HV	1.80E-10	0.00	1.80E-10	0.00	1.80E-10	0.00
	SP	1.80E-10	0.00	1.80E-10	0.00	1.80E-10	0.00
	GD	1.80E-10	1.00	1.80E-10	1.00	1.80E-10	1,00

Fonte – Elaborado pelo autor

NSGA-II iniciada na análise anterior. Desta vez o NSGA-II foi melhor apenas em SP quando comparado o IBEA, além de ser estatisticamente semelhante ao MOCell em GD. Quando comparado o IBEA ao MOCell seguimos o padrão presente em todos os *datasets*, HV sem diferença estatística, MOCell superior em SP e IBEA em GD.

Tabela 5 – Resultados **Dataset-4** dos testes Wilcoxon (WC) e Vargha-Delaney's $\hat{A}_{12}$, comparando o algoritmo linha a coluna para cada métrica de qualidade.

Algoritmos	Métricas	**Dataset-4**					
		IBEA		MOCell		NSGA-II	
		WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$	WC	$\hat{A}_{12}$
MOCell	HV	6.30E-01	**0.62**	-	-	-	-
	SP	6.00E-10	0.01	-	-	-	-
	GD	1.90E-03	0.77	-	-	-	-
NSGA-II	HV	5.30E-05	0.17	2.80E-07	0.09	-	-
	SP	1.60E-02	**0.27**	4.00E-10	0.99	-	-
	GD	6.30E-03	0.75	1.00E+00	**0.47**	-	-
Random	HV	1.80E-10	0.00	1.80E-10	0.00	1.80E-10	0.00
	SP	1.80E-10	0.00	1.80E-10	0.00	1.80E-10	0.00
	GD	1.80E-10	1.00	1.80E-10	1.00	1.80E-10	1.00

Fonte – Elaborado pelo autor

Após analisar todas os *datasets* percebe-se que nas instâncias reais, com menos tarefas e empregados, o NSGA-II tem melhores resultados. Está técnica foi quase sempre superior em *Hypervolume* (HV) e *Generational Distance* (GD), apesar de suas soluções não

serem tão distribuídas como as de seus concorrentes, sendo assim inferior na métrica *Spread* (SP), este fato não foi suficiente para diminuir seu desempenho. Porém quando analisamos instâncias maiores como as artificiais, o NSGA-II perde força. Pode-se perceber isso no *dataset-3*, onde ele até consegue manter suas frentes próximo a frente real, porém sua má distribuição representada pelo SP acaba afetando também seu desempenho em HV.

As técnicas MOCell e IBEA por sua vez se mostram regulares, em todas as instâncias são estatisticamente semelhantes no quesito *Hypervolume* (HV). No *dataset-1* MOCell foi superior em GD, enquanto o IBEA superou seu rival em SP. Porém a partir do *dataset-2* o desempenho se inverteu e seguiu assim nos *datasets* restantes, com MOCell superior em SP, o IBEA em GD. Dado esse padrão das duas técnicas nas instâncias reais as duas ficaram atrás do NSGA-II, porém com a queda deste nas instâncias artificias os dois conseguem melhor desempenho.

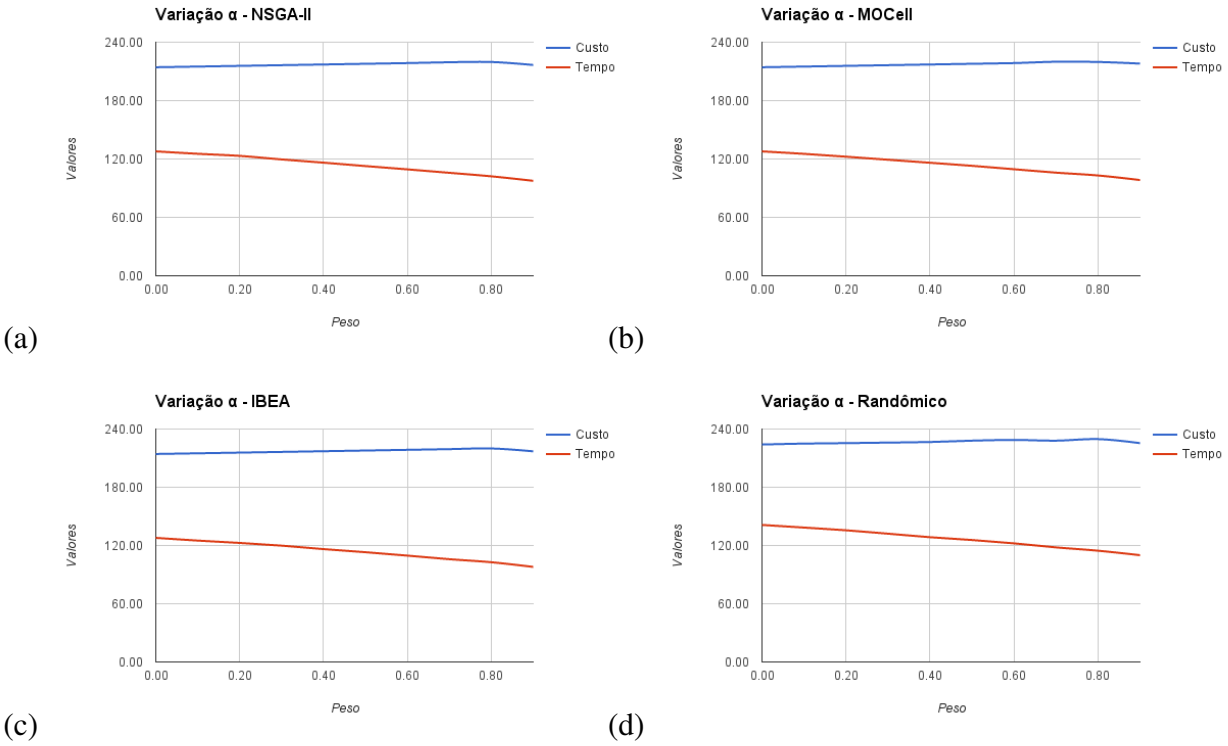
5.3.2 QP2 - Qual a efetiva influência do fator α nas soluções encontradas?

A análise feita tanto nessa parte do estudo empírico como na próxima tem como objetivo medir a influência dos pesos α e β, e consequentemente da eficiência e similaridade, nos objetivos custo e tempo. Para isso utiliza-se configurações que dificilmente serão utilizadas no mundo real, como um peso de 0.8 ou 0.9 em algum dos fatores, porém são importantes para vermos que de fato os fatores influenciam as soluções encontradas. A respeito apenas desta questão de pesquisa, é valido lembrar que a eficiência influência nos objetivos tanto incrementando como decrementando a variável TEH de uma tarefa. Para esta análise o peso β foi fixado em 0.5, e o α variado entre 0 e 0.9.

A Figura 3 mostra os gráficos para todos as técnicas utilizadas no *dataset-1*. Como pode ser visto, apesar de pequena em todos os algoritmos o tempo é reduzido com o aumento do peso. O custo por sua vez tem um pequeno incremento, esse fato pode ser explicado pelo fato de que empregados mais produtivos tendem a ser mais caros. Um segundo fator importante é que de todos as instancias o *dataset-1* é a que tem maior proporção de tarefas por empregado, diminuindo assim a margem para um aumento de produtividade.

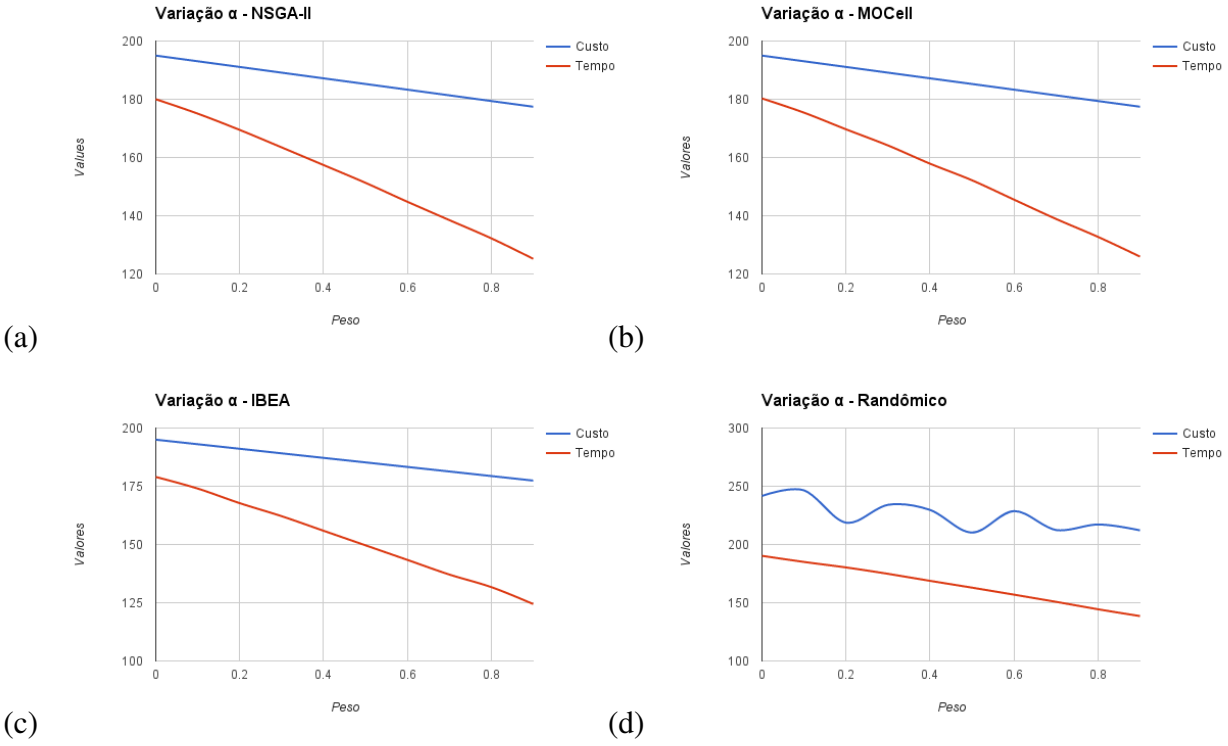
Quando analisa-se o *dataset-2* com uma proporção menor de tarefas por empregado a margem para aumento de produtividade é maior. As três meta-heurísticas tiveram desempenhos semelhantes e conseguiram diminuir, em média, 9% em custo e 30% em tempo. Novamente o objetivo custo é sacrificado para que o tempo tenha um decremento significativo. A Figura 4

Figura 3 – Variação α **Dataset-1**



Fonte – Elaborado pelo autor

Figura 4 – Variação α **Dataset-2**

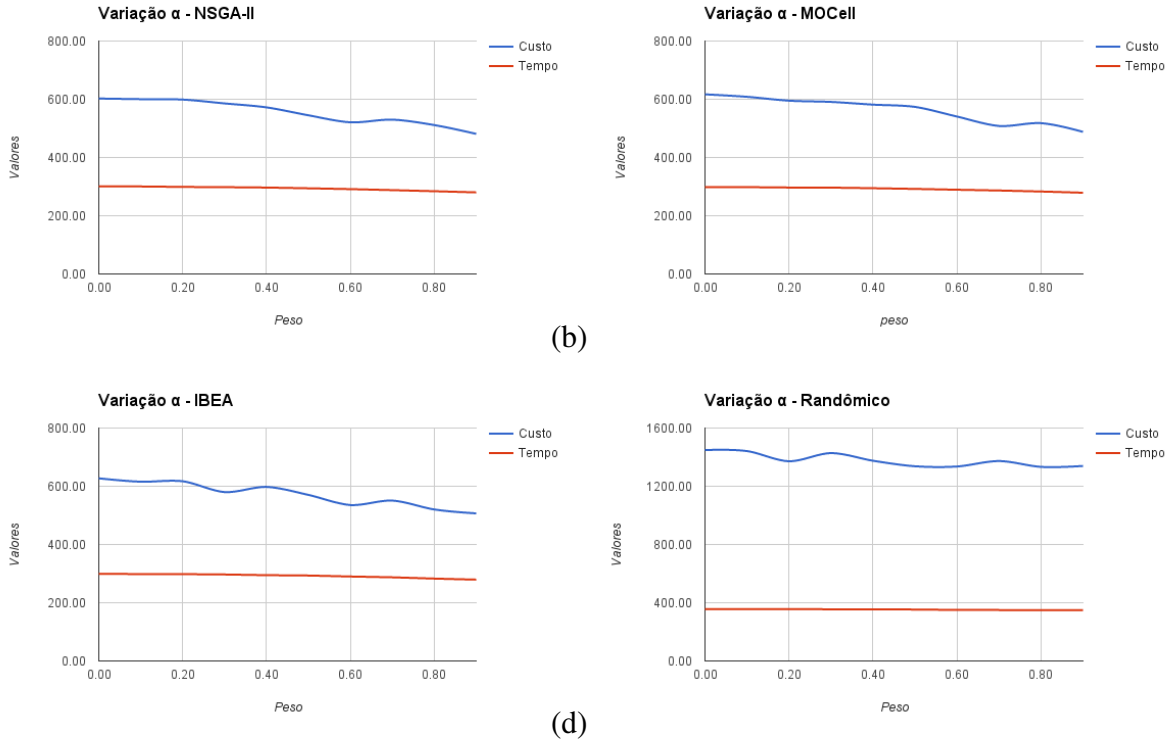


Fonte – Elaborado pelo autor

mostra essas informações.

Pode ser visto na Figura 5 o comportamento no *dataset-3*, a primeira instância artificial, difere das duas primeiras. A diminuição no custo foi maior do que em tempo, em média 20% nas meta-heurísticas, o tempo decrementou apenas 6%. Algo que chama atenção é a oscilação que acontece em determinado momento.

Figura 5 – Variação α **Dataset-3**

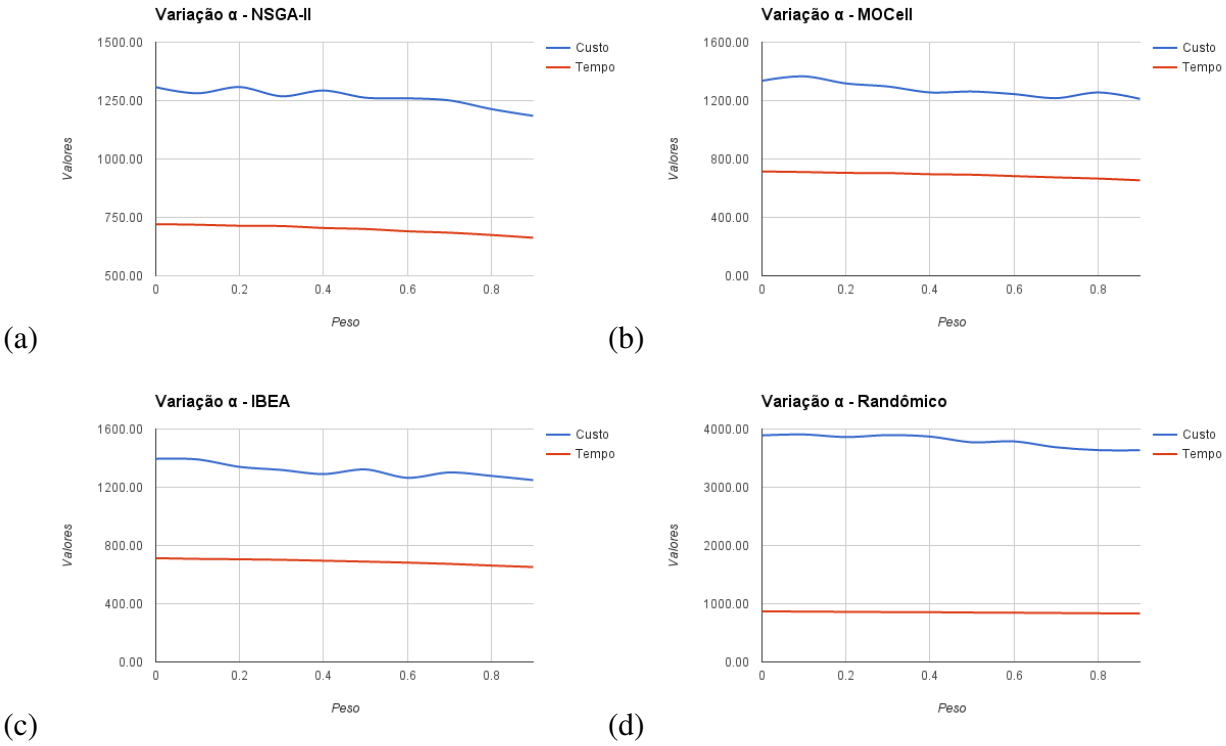


Fonte – Elaborado pelo autor

Assim como na primeira instância artificial, considerando o *dataset-4* as técnicas de busca tiveram melhor êxito na minimização do objetivo custo, porém dessa vez com uma diferença apenas ligeiramente superior. Em relação ao custo houve uma diminuição em média de 9%, enquanto em tempo de 8%, vide Figura 6.

Assim como na **QP1** o comportamento das técnicas se diferenciaram nas instâncias reais e artificiais. Neste caso os algoritmos tiveram mais sucesso em minimizar o tempo das instâncias reais, enquanto nas artificiais o êxito foi melhor na minimização do custo. Tal comportamento pode ser elucidado analisando os valores das horas estimadas nas duas classes de instâncias. Nos *datasets* reais, a variável TEH possuem média bem mais alta do que nas artificias. Consequentemente o desconto em uma quantidade de horas alta, será mais facilmente percebida.

Figura 6 – Variação α **Dataset-4**



Fonte – Elaborado pelo autor

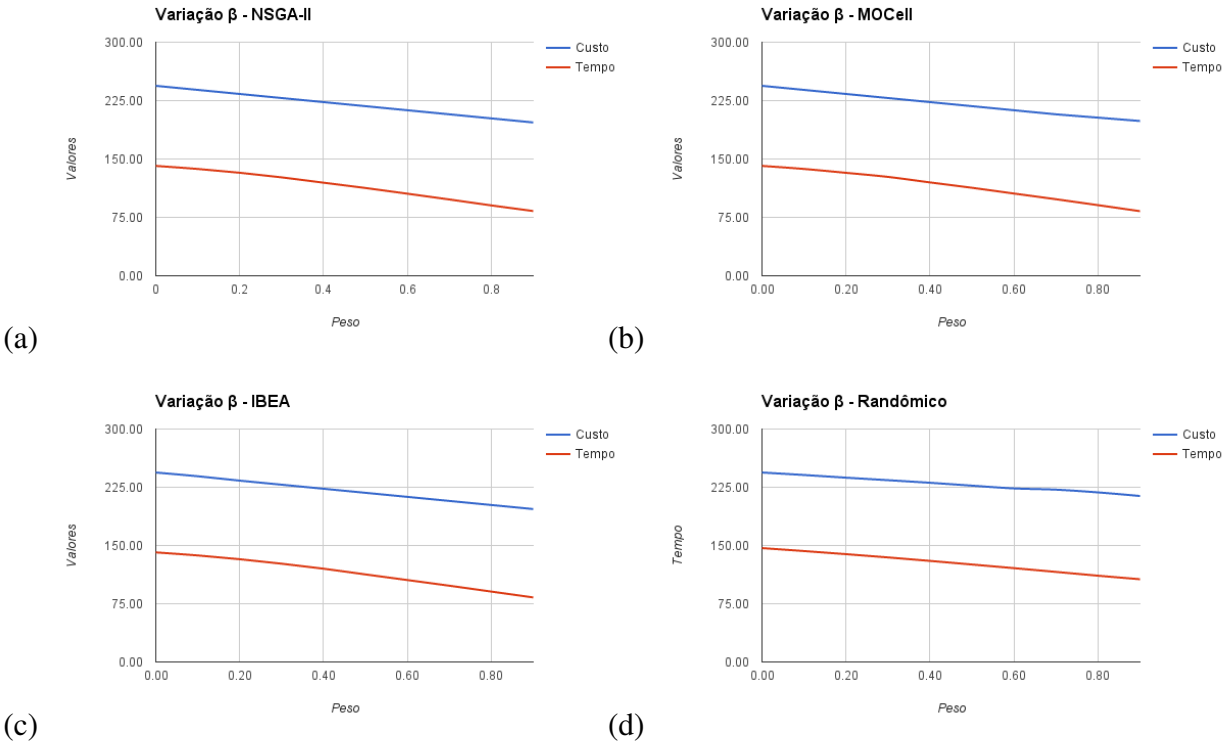
5.3.3 QP3 - Qual a efetiva influência do fator β nas soluções encontradas?

Apesar da importância de realizar uma alocação adequada com as habilidades de cada funcionário e de analisar a influência deste quesito nas soluções, o foco deste trabalho se encontra na similaridade. E é justamente o que será analisado nesta parte do estudo. Importante lembrar que o peso β em nenhuma hipótese é capaz de aumentar a variável TEH de uma tarefa, e portanto é impossível que a similaridade incremente os valores de custo e tempo. Para esta questão de pesquisa as técnicas de busca foram configuradas com o valor de β variando entre 0 e 0.9, enquanto o α foi fixado em 0.5.

A Figura 7 mostra os gráficos para cada algoritmo no *dataset-1*, como pode ser visto houve decréscimo nos dois objetivos de forma quase uniforme em todos eles, média de 20% em custo e 40% em tempo. Analisando 8 vemos nas meta-heurísticas comportamento semelhante na segunda instância, porém a busca randômica destoa dos demais, inclusive quando comparado a busca randômica da primeira.

A oscilação dos valores não incomum, dado a natureza aleatória das soluções geradas, a particularidade se encontra com o β configurado em 0.1. Neste caso há um aumento da variável custo, isso se dá pelo fato da configuração do α está fixado em 0.5, sendo assim nas primeiras

Figura 7 – Variação β **Dataset-1**



Fonte – Elaborado pelo autor

variâncias do β sua influencia nos valores de custo e tempo é menor. Uma análise mais detalhada mostra que a porcentagem de decremento dos valores dos objetivos é maior a partir da configuração 0.4.

As Figuras 9 e 10 mostram os gráficos referentes as instâncias artificias. Seguindo a tendência das investigações anteriores podemos perceber quanto maior a instância, maior a influencia da similaridade na minimização do custo. Chegando a 90% em no *dataset-3* e 82% no *dataset-4*.

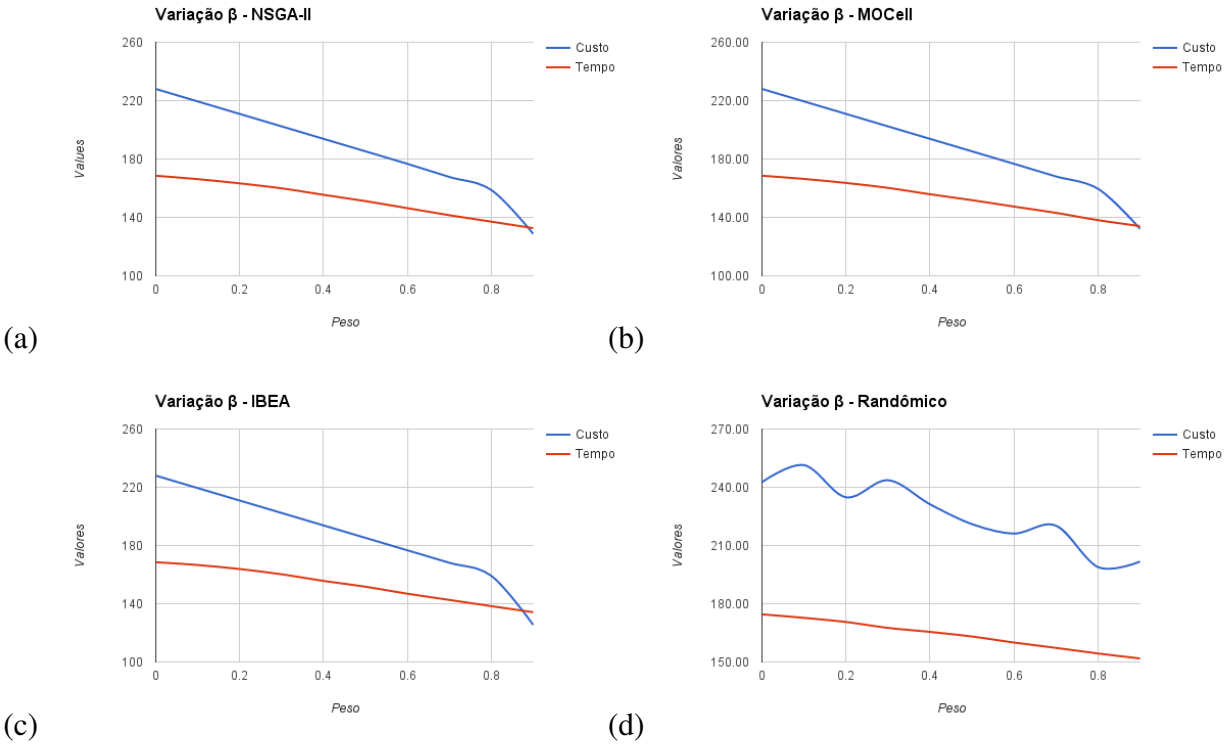
Analisando racionalmente pode-se contestar as porcentagens apresentadas até, pois a utilização de um dos pesos configurado em 0.8 incoerente. Porém o Quadro 2 mostra os ganhos com a utilização da similaridade configurada em 0.5, algo próximo da realidade.

Quadro 2 – Porcentagens médias minimizadas dos objetivos com a utilização do fator de similaridade β configurado em 0.5.

	Minimização Custo	**Minimização Tempo**
Dataset-1	11%	8%
Dataset-2	19%	10%
Dataset-3	39%	35%
Dataset-4	43%	35%

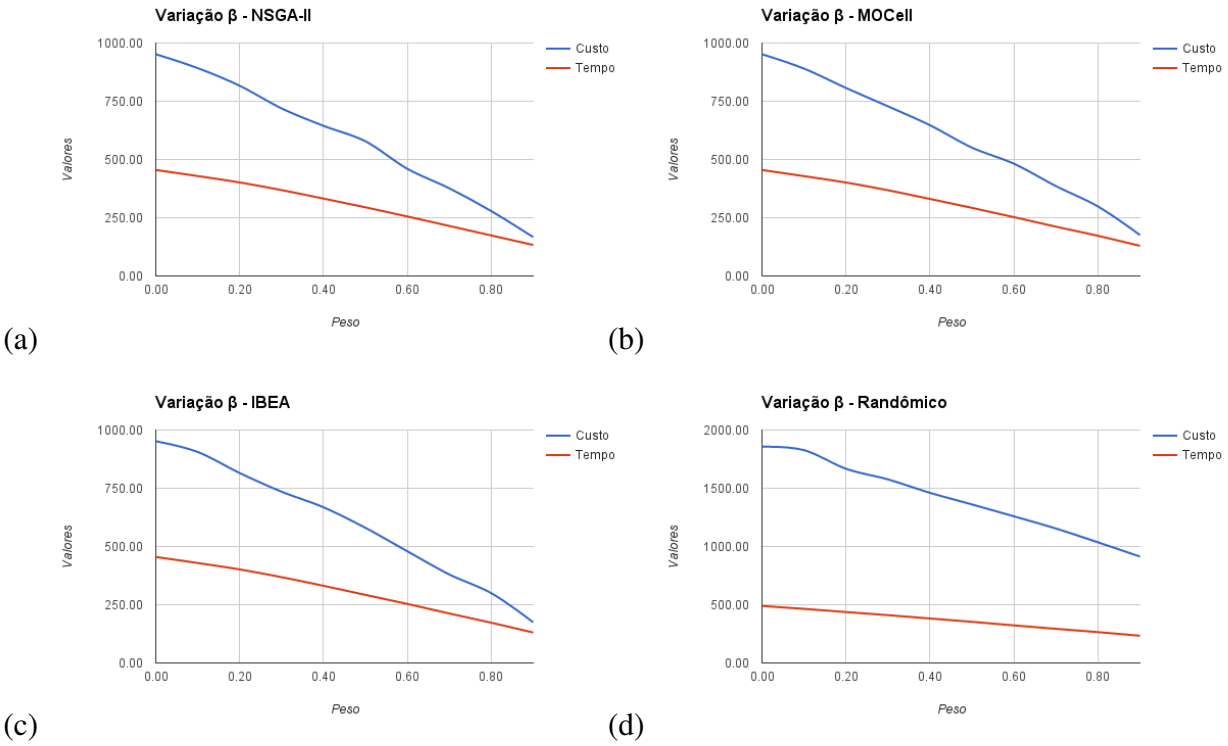
Fonte – Elaborado pelo autor

Figura 8 – Variação β **Dataset-2**



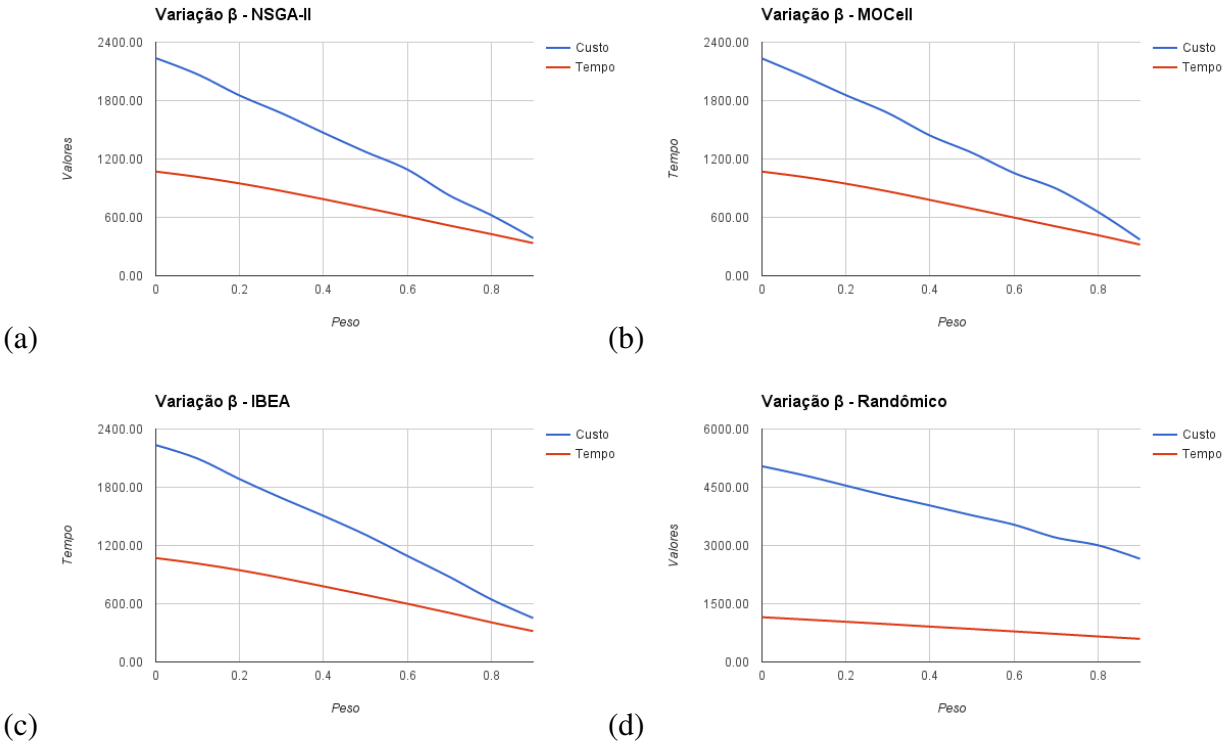
Fonte – Elaborado pelo autor

Figura 9 – Variação β **Dataset-3**



Fonte – Elaborado pelo autor

Figura 10 – Variação β **Dataset-4**



Fonte – Elaborado pelo autor

5.4 CONCLUSÕES DO CAPÍTULO

Neste capítulo foi discutido o estudo empírico realizado para validar a abordagem proposta neste trabalho. Este estudo foi conduzido orientado por três questões de pesquisa, a primeira diz respeito ao desempenho das técnicas de busca escolhidas, e as duas restantes aos fatores de eficiência e similaridade.

Com os resultados coletados foi possível constatar que todas as técnicas utilizadas conseguiram encontrar boas soluções para o problema. Nas instâncias artificias o NSGA-II obteve melhor desempenho, que obteve melhores resultados nas métricas *Hypervolume* (HV) e *Generational Distance* (GD). Porém, com o aumento das instâncias, o NSGA-II perdeu desempenho, consequentemente nas instâncias artificias foi superado pelo MOCell e o IBEA. O primeiro foi melhor em dispersão, medido pela métrica *Spread* (SP), e o segundo em *Generational Distance* (GD) que mede a convergência. No quesito HV, os dois foram estatisticamente iguais.

Ainda foi possível perceber que os fatores eficiência e similaridade, representados pelos pesos α e β, de fato influenciam nos objetivos custo e tempo. Esta investigação mostrou que a utilização dos mesmos mostra-se útil na busca da otimização dos recursos humanos.

6 CONSIDERAÇÕES FINAIS

A alocação de recursos humanos é um componente crucial no gerenciamento de um projeto de software. Com base nisso este trabalho propôs uma abordagem multiobjetivo para o Problema da Alocação de Recursos para Planejamento de Release de Software, com objetivo de distribuir um conjunto de atividades compatíveis com as habilidades de cada empregado e que sejam similares entre si, visando aumentar a produtividade do empregado, e assim diminuir o tempo e o custo do projeto.

Por meio do estudo empírico realizado foi constatado que para instâncias menores a melhor opção de técnica de busca é o *Non-dominated Sorting Genetic Algorithm-II* (NSGA-II), pois possui convergência consideravelmente melhor que seus concorrentes segundo a métrica GD, e apesar de sua baixa dispersão consegue ser melhor também em HV. Pôde-se também corroborar a utilização dos fatores de eficiência e similaridade como guias para calculo do tempo e do custo de uma alocação. Sendo influenciadoras não apenas em soluções extremas, mas também em cenários próximos da realidade.

De forma geral, foi apresentado uma nova abordagem que foi provada eficiente para resolução do RASORP, além de introduzir uma nova medida de qualidade para o problema, a similaridade

6.1 CONTRIBUIÇÕES

As principais contribuições produzidas por esse trabalho são:

(a) **Formulação Matemática:** Neste trabalho foi proposta uma formulação matemática inédita para o problema RASORP;

(b) **Nova métrica para o RASORP:** Foi também introduzido um novo parâmetro de qualidade para soluções do RASORP, a similaridade;

(c) **Publicação:** Tendo em vista o ineditismo da formulação proposta neste trabalho, foi produzido um artigo, chamado "Human resource allocation in agile software projects based on task similarities", e submetido para a trilha de estudante do 8^o Symposium on Search-Based Software Engineering (SSBSE). O referido artigo foi aceito para publicação e será apresentado em outubro de 2016 na cidade Railegh, EUA.

6.2 LIMITAÇÕES

Apesar de entender que a pesquisa trouxe bons frutos, e necessário reconhecer suas limitações. A principal delas está na dificuldade de determinar os valores de similaridade entre requisitos, somado ao fato de que todas as variáveis presentes nas tarefas e empregados necessitam ser estimadas. Outra limitação que pode ser apontada é que pela complexidade do problema, outras variáveis e métricas poderiam ser utilizadas, que possivelmente melhorariam processo de alocação.

6.3 TRABALHOS FUTUROS

Após comprovada a eficiência da abordagem, algumas evoluções são propostas com o objetivo de diminuir suas limitações e/ou aprimorar o processo de alocação de recursos humanos:

(a) Propôr uma estrategia automática para determinar a similaridade entre as tarefas;
(b) Considerar a participação de vários times trabalhando no mesmo projeto. Realizando a distribuição dos empregados e as tarefas entre os times, e posteriormente realizar o processo de alocação.

REFERÊNCIAS

ABRAHAM, A.; JAIN, L. Evolutionary multiobjective optimization. In: **Evolutionary Multi-objective Optimization**. [S.l.]: Springer, 2005. p. 1–6.

ACUNA, S. T.; JURISTO, N.; MORENO, A. M. Emphasizing human capabilities in software development. **Software, IEEE**, IEEE, v. 23, n. 2, p. 94–101, 2006.

ALBA, E.; DORRONSORO, B. **Cellular genetic algorithms**. [S.l.]: Springer Science & Business Media, 2009. v. 42.

ANTONIOL, G.; PENTA, M. D.; HARMAN, M. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In: IEEE. **Software Metrics, 2004. Proceedings. 10th International Symposium on**. [S.l.], 2004. p. 172–183.

ANTONIOL, G.; PENTA, M. D.; HARMAN, M. Search-based techniques for optimizing software project resource allocation. In: SPRINGER. **Genetic and Evolutionary Computation Conference**. [S.l.], 2004. p. 1425–1426.

ANTONIOL, G.; PENTA, M. D.; HARMAN, M. Search-based techniques applied to optimization of project planning for a massive maintenance project. In: IEEE. **21st IEEE International Conference on Software Maintenance (ICSM'05)**. [S.l.], 2005. p. 240–249.

ARCURI, A.; BRIAND, L. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: IEEE. **2011 33rd International Conference on Software Engineering (ICSE)**. [S.l.], 2011. p. 1–10.

ARCURI, A.; BRIAND, L. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. **Software Testing, Verification and Reliability**, Wiley Online Library, v. 24, n. 3, p. 219–250, 2014.

BHATTI, M. A. **Practical Optimization Methods: With Mathematica® Applications**. [S.l.]: Springer Science & Business Media, 2012.

BIBI, N.; AHSAN, A.; ANWAR, Z. Project resource allocation optimization using search based software engineering—a framework. In: IEEE. **Digital Information Management (ICDIM), 2014 Ninth International Conference on**. [S.l.], 2014. p. 226–229.

BIBI, N.; ANWAR, Z.; AHSAN, A. Comparison of search-based software engineering algorithms for resource allocation optimization. **Journal of Intelligent Systems**.

CHEN, W.-N.; ZHANG, J. Ant colony optimization for software project scheduling and staffing with an event-based scheduler. **Software Engineering, IEEE Transactions on**, IEEE, v. 39, n. 1, p. 1–17, 2013.

COELLO, C. A. C.; VELDHUIZEN, D. A. V.; LAMONT, G. B. **Evolutionary algorithms for solving multi-objective problems**. [S.l.]: Springer, 2002. v. 242.

DAHLSTEDT, Å. G.; PERSSON, A. Requirements interdependencies: state of the art and future challenges. In: **Engineering and managing software requirements**. [S.l.]: Springer, 2005. p. 95–116.

DEB, K. **Multi-objective optimization using evolutionary algorithms**. [S.l.]: John Wiley & Sons, 2001. v. 16.

DEB, K.; AGRAWAL, S.; PRATAP, A.; MEYARIVAN, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: SPRINGER. **International Conference on Parallel Problem Solving From Nature**. [S.l.], 2000. p. 849–858.

DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE transactions on evolutionary computation**, IEEE, v. 6, n. 2, p. 182–197, 2002.

DEB, K.; SINDHYA, K.; HAKANEN, J. Multi-objective optimization. In: **Decision Sciences: Theory and Practice**. [S.l.]: CRC Press, 2016. p. 145–184.

DURILLO, J. J.; NEBRO, A. J. jmetal: A java framework for multi-objective optimization. **Advances in Engineering Software**, Elsevier, v. 42, n. 10, p. 760–771, 2011.

EIBEN, A. E.; SMITH, J. E. *et al.* **Introduction to evolutionary computing**. [S.l.]: Springer, 2003. v. 53.

FOWLER, M.; HIGHSMITH, J. The agile manifesto. **Software Development**, [San Francisco, CA: Miller Freeman, Inc., 1993-, v. 9, n. 8, p. 28–35, 2001.

GASPAR-CUNHA, A.; TAKAHASHI, R.; ANTUNES, C. H. **Manual de computação evolutiva e metaheurística**. [S.l.]: Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.

HARMAN, M. The current state and future of search based software engineering. In: IEEE COMPUTER SOCIETY. **2007 Future of Software Engineering**. [S.l.], 2007. p. 342–357.

HARMAN, M.; CLARK, J. Metrics are fitness functions too. In: IEEE. **Software Metrics, 2004. Proceedings. 10th International Symposium on**. [S.l.], 2004. p. 58–69.

HARMAN, M.; JONES, B. F. Search-based software engineering. **Information and software Technology**, Elsevier, v. 43, n. 14, p. 833–839, 2001.

HARMAN, M.; MCMINN, P.; SOUZA, J. T. D.; YOO, S. Search based software engineering: Techniques, taxonomy, tutorial. In: **Empirical software engineering and verification**. [S.l.]: Springer, 2012. p. 1–59.

HOLLAND, J. H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.** [S.l.]: U Michigan Press, 1975.

KANG, D.; JUNG, J.; BAE, D.-H. Constraint-based human resource allocation in software projects. **Software: Practice and Experience**, Wiley Online Library, v. 41, n. 5, p. 551–577, 2011.

LINDEN, R. **Algoritmos genéticos (2a ediçao)**. [S.l.]: Brasport, 2008.

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**, Wiley Online Library, v. 24, n. 7, p. 726–746, 2009.

NGO-THE, A.; RUHE, G. Optimized resource allocation for software release planning. **Software Engineering, IEEE Transactions on**, IEEE, v. 35, n. 1, p. 109–123, 2009.

PENTA, M. D.; HARMAN, M.; ANTONIOL, G. The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study. **Software: Practice and Experience**, Wiley Online Library, v. 41, n. 5, p. 495–519, 2011.

SCHWABER, K. **Agile project management with Scrum**. [S.l.]: Microsoft press, 2004.

SOMMERVILLE, I. **Software Engineering**. 9. ed. [S.l.]: Addison-Wesley, 2010.

SRINIVAS, N.; DEB, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary computation**, MIT Press, v. 2, n. 3, p. 221–248, 1994.

THIELE, L.; MIETTINEN, K.; KORHONEN, P. J.; MOLINA, J. A preference-based evolutionary algorithm for multi-objective optimization. **Evolutionary computation**, MIT Press, v. 17, n. 3, p. 411–436, 2009.

TSAI, H.-T.; MOSKOWITZ, H.; LEE, L.-H. Human resource selection for software development projects using taguchi's parameter design. **European Journal of Operational Research**, Elsevier, v. 151, n. 1, p. 167–180, 2003.

ZHANG, Y. **Multi-Objective Search-based Requirements Selection and Optimisation**. [S.l.]: University of London, 2010.

ZITZLER, E.; KÜNZLI, S. Indicator-based selection in multiobjective search. In: SPRINGER. **International Conference on Parallel Problem Solving from Nature**. [S.l.], 2004. p. 832–842.