UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

MESTRADO ACADÊMICO EM COMPUTAÇÃO

CAETANO VIEIRA NETO SEGUNDO

UMA SIMULAÇÃO MULTIAGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE BASEADA NO TRUCK FACTOR

QUIXADÁ

2022

CAETANO VIEIRA NETO SEGUNDO

UMA SIMULAÇÃO MULTIAGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE BASEADA NO TRUCK FACTOR

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Marcos Antonio de Oliveira

Coorientador: Prof. Dr. Enyo José Tavares Gonçalves

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V714s Vieira Neto Segundo, Caetano.
Uma Simulação multiagente para alocação de tarefas em projetos de software baseada no truck factor / Caetano Vieira Neto Segundo. – 2022.
92 f.

Dissertação (mestrado) – Universidade Federal do Ceará, Campus de Quixadá, Programa de Pós-Graduação em Computação, Quixadá, 2022.
Orientação: Prof. Dr. Marcos Antonio de Oliveira.
Coorientação: Prof. Dr. Enyo José Tavares Gonçalves.

1. Sistemas Multiagentes. 2. Algoritmos Genéticos. 3. Software - Desenvolvimento. I. Título.

CDD 005

CAETANO VIEIRA NETO SEGUNDO

UMA SIMULAÇÃO MULTIAGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE BASEADA NO TRUCK FACTOR

Dissertação apresentada ao Curso de Mestrado Acadêmico em Computação do Programa de Pós-Graduação em Computação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em computação. Área de Concentração: Ciência da Computação

Aprovada em: __/__/__

BANCA EXAMINADORA

Prof. Dr. Marcos Antonio de Oliveira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Enyo José Tavares Gonçalves (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Lincoln Souza Rocha
Universidade Federal do Ceará (UFC)

Prof. Dr. Davi Romero de Vasconcelos
Universidade Federal do Ceará (UFC)

Prof. Dr. Gustavo Augusto Lima de Campos
Universidade Estadual do Ceará (UECE)

Esta pesquisa é dedicada para a minha mãe Ana Maria Gomes de Lima, que apesar das dificuldades, sempre batalhou para que eu e meus irmãos pudéssemos ter as melhores oportunidades. Você é um exemplo de vida para nós. Sou abençoado por ser seu filho. Obrigado por tudo.

AGRADECIMENTOS

Toda a trilha percorrida durante esse tempo, foi de muito aprendizado. Para a conclusão desta pesquisa, tenho o apoio e participação de muitas pessoas, e por todos sou grato.

Em primeiro lugar, não posso deixar de agradecer ao meu orientador, Prof. Marcos Antônio de Oliveira e co-orientador e Prof. Enyo José Tavares, por toda a sua paciência e seu apoio, esses foram essenciais para a conclusão da pesquisa. Muito obrigado por nossas conversas e ensinamentos.

Agradeço aos meus pais, Antônio Vieira da Costa e Ana Maria Gomes de Lima, como também a toda minha família, em especial ao meu irmão Rubbens Anttônio de Lima Vieira, e minha irmã Jamille de Lima Vieira, por todo o seu apoio e suporte.

Obrigado a todos os meus colegas do mestrado, Antônio, Gilberto, Warlles, Bergh, José, Rogério, Júnior, Leonardo e Wellington, pessoas com quem dividi durante esse tempo momentos de dificuldades,mas também bons momentos de convivência e comemoração.

Obrigado em especial, para uma pessoa que conheci durante essa caminhada, com quem pude trocar vivências, ter aprendizados e me tornar amigo, Isaac Rahel Martim Oliveira. Obrigado por todo o seu suporte. Aplaudo o seu sucesso.

Agradeço a toda família Mourão, em especial a Laninha e a Vilanir, pessoas muito especiais que tive o prazer de conhecer melhor durante essa caminhada, que puderam me receber e me fazer sentir parte da família.

Em conclusão, obrigado Joyce Maria Honório Rodrigues (Joy). Obrigado por ser minha companheira de vida, amo partilhar a vida boa com você. Sou grato por ter você ao meu lado, pela sua paciência, compreensão e ajuda prestada em todos os momentos, seu apoio sempre foi fundamental para mim.

A todos, sinceramente obrigado.

“Se eu vi mais longe, foi por estar sobre ombros de gigantes.”

(Isaac Newton)

RESUMO

No desenvolvimento de um projeto de software existem diversas etapas que devem ser realizadas para que o mesmo seja concluído com êxito. Alocação de tarefas é uma dessas etapas. Ao longo dos anos tem-se pesquisado sobre técnicas que visam otimizar tal processo a fim de minimizar custo e tempo de desenvolvimento. No entanto, o *Truck Factor* (TF) é uma métrica que pode determinar risco a um projeto e também deve ser levado em consideração no momento de distribuir tarefas entre os membros da equipe. O TF diz respeito à distribuição de conhecimento sobre o projeto entre os membros da equipe de desenvolvimento. Dessa forma, faz com que não haja concentração de conhecimento apenas por uma parte da equipe, tal temática é de extrema relevância, visto que com a globalização e demanda do mercado de TI, a rotatividade de seus profissionais tornou-se comum. Sabe-se que alocação de recursos em times de software não possui apenas solução exata, visto que é um problema NP-difícil e, a partir disso, ao longo dos anos técnicas de *Search Based Software Engineering* (SBSE), que possui uma diversidade de algoritmo de otimização, como por exemplo algoritmos genéticos, que têm sido aplicados em diversas pesquisas a fim de resolver tal problemática. Em ambientes multiagentes, entende-se que existe uma determinada quantidade de agentes que têm percepção e se comunicam a fim de atingir seus objetivos. Pesquisas na área de multiagentes têm se utilizado de ambientes simulados a fim validar suas pesquisas, visto que a modelagem e simulação devem contemplar as principais variáveis de um ambiente real. Portanto, este trabalho propõe construir uma simulação multiagente, que utiliza técnicas de SBSE, com o objetivo de minimizar os impactos causados pelo Truck Factor em uma equipe de desenvolvimento de software. Nas simulações, modelamos configurações diferentes para as equipes de desenvolvimento de software. Nossos resultados mostram que a abordagem proposta minimiza os impactos causados pela alocação de tarefas em equipes de desenvolvimento de software quando consideramos a métrica TF durante a atribuição de tarefas.

Palavras-chave: Alocação de Tarefas; Simulação Multiagente; Search Based Software Engineering; Algoritmo Genético

ABSTRACT

In the development of software, a project several steps must be carried out for it to be completed. Task allocation is one such step. Over the years, techniques have been researched to optimize this process to minimize cost and development time. However, it is believed that the Truck Factor (TF) is a metric that can determine the risk to a project and should also be taken into account when distributing tasks among team members. The TF concerns the distribution of knowledge about the project among the members of the development team, thus ensuring that there is no concentration of knowledge by only one part of the team, this theme is extremely relevant since, with globalization and demand from the IT market, the rotation of its professionals has become common. It is known that resource allocation in software teams does not have only an exact solution, since it is an Np-hard problem and, from that, over the years Search-Based Software Engineering (SBSE) techniques, which have a diversity optimization algorithms, such as genetic algorithms, have been applied in several types of research to solve this problem. In multi-agent environments, it is understood that there is a certain amount of agents that have perception and communicate to achieve their goals. Research in the area of multi-agents has used simulated environments to validate their research since the modeling and simulation must consider the main variables of a real environment. Therefore, this work proposes to build a multi-agent simulation, which uses SBSE techniques, to minimize the impacts caused by Truck Factor in a software development team. In the simulations, we model different configurations for the software development teams. Our results show that the proposed approach minimizes the impacts caused by task allocation in software development teams when we consider the TF metric during task assignment.

Keywords: Task Allocation; Multiagent Simulation; Search Based Software Engineering; Genetic Algorithm

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

SBA	Simulação Baseada em Agentes
SBSE	Search Based Software Engineering
PO	Product Owner
SMA	Sistemas Multiagentes
TF	Truck Factor
ES	Engenharia de Software
DOA	Degree of Authorsghip
ABM	Agent Based Modeling

SUMÁRIO

1 INTRODUÇÃO

Essa dissertação tem como objetivo contribuir na área de alocação de tarefas em projetos de software. Por meio de uma abordagem de Simulação Baseada em Agentes(SBA) que utiliza técnicas de Search-Based Software Engineering (SBSE) na alocação de tarefas em times de desenvolvimento de software.

Este capítulo introduz acerca do contexto de alocação de recursos em times de software, em seguida apresenta a motivação para a elaboração da dissertação e seus objetivos gerais e específicos, por fim é apresentada a estrutura organizacional desta dissertação.

1.1 Contextualização

No desenvolvimento ágil com Scrum, o *Product Owner* (PO) tem o papel de definir o que será realizado, além de conhecer sobre todas as regras de negócio do sistema (RUBIN, 2012). As equipes de desenvolvimento são compostas de distintas características de conhecimento, habilidades e personalidades. A alocação dessas tarefas se torna uma atividade crítica dentro do processo, visto que o sucesso do projeto depende de como são divididas as atividades que devem ser realizadas no projeto (BIBI *et al.*, 2014). Portanto, entende-se que a alocação dos recursos humanos tem impacto direto no sucesso ou na falha do projeto.

A alocação de tarefas é um problema complexo de se solucionar, visto que existem diferentes soluções para o mesmo problema. Outra característica importante nessa temática, é que em muitas das vezes há mais de um objetivo a ser atingido, os mais comuns são a minimização do custo e do tempo. Esses objetivos são de extrema importância, entendendo que gerencialmente o projeto torna-se mais viável quando se tem um time eficiente com um custo adequado.

Outro fator que deve ser levado em consideração é o *Truck Factor*, este parâmetro diz respeito ao quanto o projeto é dependente em relação ao conhecimento da equipe dentro do ambiente de desenvolvimento (AVELINO *et al.*, 2016). O cenário ideal ocorre quando o conhecimento que a equipe possui sobre o projeto seja o mais distribuído possível, para que em casos de trocas dos seus membros, o projeto continue em seu ritmo natural de desenvolvimento.

A distribuição do conhecimento é fundamental, através de uma análise sobre a rotatividade em projetos de software, vemos que é comum a troca de membros da equipe, isso ocorre por diferentes fatores. Autores ao redor do mundo já estudam essas mudanças de equipes como também os impactos e fatores que causam a rotatividade (HILTON; BEGEL, 2018) e

(FAROOQ *et al.*, 2022). Um dado, apresentado por (PEE *et al.*, 2014), mostra que em 2012, cerca de 20% a 28% dos projetos de software falharam, isso por diferentes motivos, portanto, a rotatividade da equipe se mostrou um risco significante à continuidade do projeto. De modo mais recente abordado por (MOHAMAD *et al.*, 2021) em 2018, o setor da indústria da TI teve uma rotatividade de 10.9%, e dentro do setor de TI, os times de softwares foram os mais afetados.

Portanto, entendendo que essa saída e entrada de novos membros é praticamente inevitável muitas das vezes. É fundamental elaborar uma abordagem que analise não somente os parâmetros tradicionais (custo e tempo) que já são amplamente considerados em pesquisas, mas também a inclusão da análise do *Truck Factor*. Tal parâmetro se mostra como um critério primordial, visto que uma boa eficiência nesse fator de risco do projeto, fornece garantias de continuidade do trabalho e distribuição balanceada do conhecimento em situações de trocas de membros nas equipes.

O problema da alocação de recursos é amplamente abordado através de Sistemas Multiagentes em diferentes contextos como: (SINGHAL; DAHIYA, 2015) , (KROTHAPALLI; DESHMUKH, 2002) e (LI *et al.*, 2021b), visto que através de um SMA é possível projetar diferentes agentes em seu ambiente que executam tarefas e trocam informações entre si a fim de atingir seus objetivos. Os agentes podem competir, cooperar ou até mesmo se organizar baseado em modelos com o propósito de obter êxito. Dentro de um mesmo ambiente, podem existir diversos agentes com diferentes papéis e características. SMA permite projetar e construir sistemas com as mesmas ideias e conceitos aplicados nas relações e hábitos sociais (ABBAS *et al.*, 2015).

Simulações envolvendo agentes para representar situações do mundo real têm sido tema recorrente de pesquisas na área de SMA, como abordado por (PETROVIĆ, 2020) e (MAHESH *et al.*, 2022). Construir modelos através da simulação permite mapear problemas do mundo real e implementá-los em um ambiente virtual, tornando possível analisar e otimizar o modelo proposto (CHUMACHENKO *et al.*, 2019). A simulação consiste em um conjunto de regras que dita como o sistema se modifica ao longo do tempo, e seu objetivo é uma melhor compreensão de um determinado evento ou até mesmo sua predição (SIEBERS; AICKELIN, 2008).

Como abordado por (BIBI *et al.*, 2014), (DEMIROVIĆ *et al.*, 2018) e (LI *et al.*, 2021a), a alocação de tarefas vem sendo pesquisada de maneira recorrente por diferentes áreas de estudo, e é classificada como um problema NP-Difícil. Por meio de Search-Based Software

Engineering (SBSE), (HARMAN; JONES, 2001) se propõe a aplicar algoritmos de otimização baseados em busca para problemas da Engenharia de Software.

Problemas da Engenharia de Software que tem como objetivo minimizar ou maximizar podem ser tratados por meio de algoritmos de busca. Ao longo dos anos, diversas pesquisas vêm solucionando problemas de alocação de tarefas por meio de SBSE. As técnicas mais comuns são: Simulated Annealing (SA), Ant Colony (ACO), Genetic Algorithm (GA), Integer Programming (IP), Hill Climbing (HC) e Redes Neurais (RN) (RÄIHÄ, 2010). Em SBSE, Algoritmos Genéticos são amplamente utilizados em diferentes contextos (TONELLA *et al.*, 2010), (PAIXAO; SOUZA, 2015) e (ARAÚJO, 2015), visto sua capacidade de fornecer uma solução ótima com um baixo custo computacional.

Tal temática se apresenta com extrema relevância, pois entendendo que esse processo de alocar recursos em times de software surge desde pequenas equipes de desenvolvimento até nas grandes empresas, e que a demanda por profissionais de tecnologias se mostra de maneira emergente em todo o mundo, e que é comum em que equipes de software durante a construção do projeto sejam alteradas com entrada e saída de membros. É fundamental mitigar os problemas causados pelo Truck Factor, entendendo que este é um parâmetro de influência no sucesso ou fracasso do projeto.

Projetar e construir simulação em sistemas multiagentes que visam solucionar tal problemática através da abordagem de SBSE podem contribuir nesta área de pesquisa, uma vez que através da simulação torna-se viável compreender, explorar e analisar os impactos causados pelo *Truck Factor* (TF) na distribuição de tarefas em diferentes equipes de software.

1.2 Objetivos

O presente trabalho tem como objetivo principal elaborar uma abordagem para solucionar o problema de alocação de tarefas em times de software, por meio da construção de um modelo baseado em simulação, ela visa minimizar os impactos causados pelo *Truck Factor*. Como objetivos específicos temos:

a) Modelar ambiente de simulação e seus respectivos agentes. Os agentes devem representar os diferentes papéis e características presentes em um time de desenvolvimento de software.
b) Modelar algoritmo baseado em SBSE a fim de encontrar a solução ótima na alocação de tarefas do ambiente multiagente.
c) Definir os cenários que serão desenvolvidos, e como se dá o processo da divisão de tarefas,

levando em consideração o *Truck Factor*.

d) Testar e analisar o comportamento da abordagem proposta em diferentes ambientes de simulação.

1.3 Metodologia do Trabalho

Esta pesquisa tem embasamento teórico por meio de artigos, livros, periódicos e conferências que estão diretamente ligadas ao tema abordado. Através desse levantamento, foi possível compreender o problema de alocação de tarefas em equipes de software, como também sobre Sistemas Multiagentes, Search-Based Software Engineering (SBSE) e *Truck Factor*.

A fim de construir nossa proposta, foi primordial construir o ambiente multiagente que representasse um cenário de times de software, visto que o ambiente de desenvolvimento possui uma metodologia de trabalho, foi preciso realizar um estudo a fim de compreender seus agentes, papéis e funções, onde a partir dessa análise foi definida a metodologia Scrum. Além disso, foi necessário levantar técnicas abordadas em SBSE. Nesta área são abordados diversos algoritmos que tem objetivo de encontrar uma solução ótima para problemas de ES, portanto para esta pesquisa foi escolhido Algoritmos Genéticos (AG). Neste trabalho, o *Truck Factor* é uma métrica utilizada com objetivo de ser maximizada, logo foi fundamental levantar técnicas relacionadas ao *Truck Factor*, onde a abordagem definida por (AVELINO *et al.*, 2016) apresentou-se mais adequada.

Um estudo foi realizado sobre ferramentas de simulação para ambientes multiagentes, através do mesmo foi feita a escolha da plataforma Netlogo (NETLOGO, 2022), uma vez que proporciona os recursos necessários para construção com os diferentes tipos de agentes no ambiente e validação da proposta deste trabalho.

Para a implementação do algoritmo genético, utilizou-se a linguagem Python (PYTHON, 2022), em conjunto com a biblioteca DEAP (DEAP, 2022), que é uma biblioteca que oferece suporte para a implementação de algoritmos evolucionários.

A validação do estudo se deu por meio de um modelo simulado, e que nele estão os parâmetros e variáveis necessárias que visam representar um ambiente real. Através do modelo de ambiente simulado, foi possível realizar os experimentos, a coleta, e uma série de análises estatísticas dos dados produzidos pelo ambiente.

1.4 Organização da Dissertação

O trabalho está organizado em cinco capítulos, incluindo a presente introdução. De maneira geral, o restante dos capítulos estão resumidos abaixo:

a) O Capítulo 2 apresenta o embasamento teórico com os principais conceitos para a compreensão desse trabalho de dissertação com os conceitos de SBSE, SMA, Gerenciamento de Equipes e *Truck Factor*.
b) O Capítulo 3 faz uma análise sobre os trabalhos relacionados utilizados como base ou comparação para este trabalho.
c) O Capítulo 4 apresenta detalhes sobre o problema abordado além de descrever toda a construção da abordagem proposta nesta pesquisa.
d) O Capítulo 5 mostra uma discussão acerca dos resultados obtidos pela pesquisa.
e) Por fim, no Capítulo 6, são apresentados as conclusões da pesquisa, bem como as contribuições realizadas e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordadas as principais técnicas e conceitos aplicadas nesta dissertação. As sessões serão divididas por área de atuação. Tem-se o objetivo de permitir que o leitor compreenda os fundamentos de cada área. Serão apresentados os conceitos sobre: Algoritmos Genéticos, Search Based Software Engineering (SBSE), Sistemas Multiagentes (SMA), Truck Factor (TF), Metodologia Ágil e times de Software.

2.1 Metodologia Ágil de Desenvolvimento

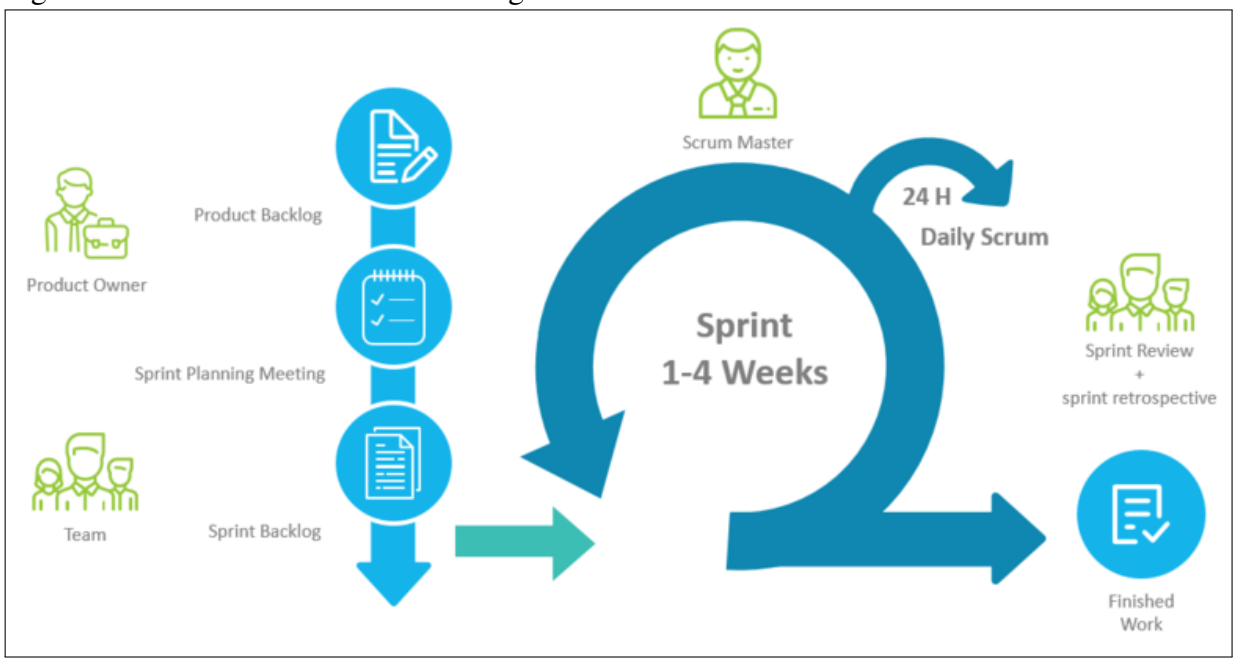
Os agentes necessitam trocar informações, atribuir tarefas e assumir diferentes papéis, com o objetivo de otimizar o processo de alocação de tarefas. Portanto, torna-se necessário utilizar uma metodologia de desenvolvimento em que os agentes devem seguir a fim de atingir seus objetivos. Este método entende que existem agentes com diferentes habilidades e funções dentro do processo de desenvolvimento. Na engenharia de software são pesquisados diversos métodos e técnicas para desenvolvimento de sistemas, em cada um desses métodos surgem equipes de software de formas diferentes e adequadas aos contexto.

Os métodos ágeis são bastantes utilizados para gerenciamento de times de software, dada a sua simplicidade e flexibilidade. Como visto em (SOMMERVILLE, 2011), estes métodos permitiram que a equipe de desenvolvimento trabalhe no software em si, e não em sua concepção e documentação. Métodos ágeis, universalmente, baseiam-se em uma abordagem incremental para a especificação, o desenvolvimento e a entrega do software. Eles são mais adequados ao desenvolvimento de aplicativos nos quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento. Destinam-se a entregar o software rapidamente aos clientes, em funcionamento, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações posteriores do sistema.

Dentre os métodos ágeis, o Scrum (SUTHERLAND, 2015) destaca-se devido a simplicidade e flexibilidade do seu processo de trabalho. Como visto na Figura 1 é apresentado o ciclo de desenvolvimento do Scrum. Nesse método o foco é na entrega contínua e ao final de cada ciclo haverá um novo incremento no sistema. No Scrum existem papéis, tarefas e artefatos.

Papéis fundamentais são: *Scrum Master* este que é responsável pela gerência do time, ajuda a todos os envolvidos a entender e abraçar os valores, princípios e práticas do Scrum; *Product Owner*, é o principal ator dentro do processo que possui liderança sobre o produto. Ele é

Figura 1 – Processo desenvolvimento ágil



Fonte: (WARCHOLINSKI, 2020)

o único responsável por decidir quais serão os recursos e funcionalidades a serem construídos e qual a ordem que devem ser realizados; *Developer Team* é o nome dado a equipe que conta com profissionais multidisciplinares, visto que são, programadores, arquitetos de software, testadores e etc.

A principal tarefa do método é a execução da *sprint*, esta que tem duração entre 1 a 4 semanas, sempre busca ao final entregar um produto tangível seja para o usuário ou para o cliente. Ou seja, no decorrer das sprints todo o sistema é construído, para além dessas atividades, há outros processos que são necessários para uma gestão eficaz das sprints, tais como: revisão da sprint, reuniões diárias, planejamento da sprint, retrospectiva da sprint.

Os artefatos gerados são: *product backlog*, este que é um documento que está constantemente em evolução. Os itens podem ser alterados e revistos pelo Product Owner por conta de mudanças nas condições de negócios; Sprint backlog, é um conjunto de atividades que são retiradas do product backlog para que sejam realizadas durante o período estabelecido da sprint.

Visto que o Scrum é adaptativo, a atribuição de tarefas pode ocorrer no planejamento ou no decorrer da própria Sprint. Desse modo, a delegação das tarefas é realizada por algum agente com essas responsabilidade, ou até mesmo o time por si só define o que cada um deve fazer. Os critérios para atribuição também não são específicos, em geral são critérios técnicos, como por exemplo a habilidade do agente em desempenhar determinada função, porém de acordo

com o objetivo que se busca, os critérios para a atribuição das tarefas podem ser adaptados.

Dessa maneira, entende-se que a aplicação de um processo ágil em uma equipe composta de agentes torna-se fundamental, pois como o scrum possui ciclos curtos a cada final de ciclo é possível obter métricas de desempenho de cada agente, para que na próxima sprint seja readequado.

2.2 Truck Factor

Como apresentado na seção anterior, a construção de um software se dá de maneira contínua e em ciclos repetitivos, até a sua conclusão que pode levar meses e até anos. Na prática, muitos softwares necessitam de constantes atualizações e de novas funcionalidades. Conclui-se então que a construção de um software é algo que consome muito tempo e recurso humano.

Ao longo do tempo, as equipes podem sofrer modificações, visto que é comum a rotatividade em equipes de software. A rotatividade em equipes de software pode gerar instabilidade no projeto e perda de desempenho, tendo em vista que as mudanças podem provocar oscilações na entrega contínua do produto. Além do desempenho, alterações podem afetar o tempo de entrega, de custo, ou até mesmo decretar o fim do projeto (PEE *et al.*, 2014).

A mudança da equipe é algo na prática inevitável, pois não há como garantir que sempre as mesmas pessoas estarão disponíveis para o desenvolvimento das tarefas. Com isso torna-se necessário analisar e compreender os impactos e riscos que tais modificações podem ocasionar. Portanto, entende-se que a rotatividade dessas equipes de software apresentam grandes riscos à sobrevivência do projeto ao longo prazo (FERREIRA *et al.*, 2016).

Uma vez que as mudanças em times de software tornam-se inevitáveis, faz-se necessário que esse problema seja estudado e minimizado. Um dos pontos chaves de todos os projetos é a gestão do conhecimento, a saída de algum membro importante da equipe, pode comprometer o desenvolvimento do projeto, visto que o restante do time tem pouco conhecimento a respeito do projeto (JABRAYILZADE *et al.*, 2022). A partir disso surgiram pesquisas que buscam compreender o conhecimento que os membros da equipe têm sobre o projeto e seus respectivos impactos.

O Truck Factor é uma medida de risco sobre o projeto, que se dá pela omissão de informações acerca do mesmo dentre seus membros. Pode ser definido como o número de pessoas que são atropeladas por um caminhão (Truck), onde tal fator apresenta impacto no desenvolvimento do projeto, visto que o atropelamento do caminhão representa a saída de

membros do time (FERREIRA *et al.*, 2017). Neste área de pesquisa, entende-se que quando membros da equipe possuem uma concentração de conhecimento técnico sobre o projeto, formam silos de conhecimento, e tal fato pode prejudicar o projeto, desde atrasos na entrega, ou até mesmo o encerramento do projeto.

Uma vez que esse problema pode surgir em qualquer equipe de software, gestores de projetos, devem compreender que o conhecimento acerca do projeto precisa ser difundido entre todos os membros da equipe. Dessa forma, é possível reduzir os impactos na equipe com a saída ou a entrada de novos membros.

Em sua essência, um baixo valor de TF indica que o sistema possui uma grande concentração de conhecimento em poucas pessoas da equipe, assim formando silos por esses indivíduos. Um alto valor de TF indica que existe equilíbrio na distribuição de conhecimento acerca do projeto (AVELINO *et al.*, 2016).

A partir de tal problemática, pesquisadores buscam calcular com base em artefatos desenvolvidos pelo time, qual seria o fator de impacto do mesmo. Em diversas outras pesquisas (ZAZWORKA *et al.*, 2010; SANDIM *et al.*, ; HANNEBAUER; GRUHN, 2014) os autores investigam, comparam e propõem novas técnicas através de algoritmos, para que se possa calcular qual é o Truck Factor de um determinado projeto.

Na abordagem proposta por (AVELINO *et al.*, 2016), o autor propõe uma abordagem gulosa para cálculo do TF. Eles atribuem um DOA (Degree Of Authorship) para cada desenvolvedor que está no projeto e fez pelo menos 1 commit no arquivo que está sendo analisado, o DOA possui 3 parâmetros, que são: 1. Primeiro autor, ele diferencia quem inicialmente criou o arquivo, 2. Quantidade de mudanças realizadas no arquivo e 3. Quantidade de mudanças realizadas no arquivo pelo restante da equipe. Uma vez que todos os valores de DOA são conhecidos, é realizado um processo de normalização, onde são definidos valores entre 0 e 1, no final o autor do arquivo é considerado aquele que possui DOA > 0.75.

Nesta abordagem, entende-se que o projeto está correndo sérios riscos de atrasos, podendo até ser descontinuado, caso a cobertura do conjunto de autores que estão atualmente esteja inferior a 50%. O resultado do Truck Factor produzido mostra a quantidade de autores que podem ser retirados do projeto, até que a cobertura não seja inferior a 50%.

No algoritmo 1 é apresentada uma implementação através de pseudocódigo. O algoritmo tem como entrada um conjunto mapeado de autores e arquivos, dessa forma é possível saber quais foram os autores que contribuíram em cada arquivo. O algoritmo inicia o tf com

Algoritmo 1: Algoritmo do Truck Factor

```
Input: Lista de Autores (A) e Arquivos
Output: Truck Factor do sistema
S ← getArquivos(A);
tf ← 0;
while A ≠ θ do
    cobertura ← getCobertura(S, A);
    if cobertura < 0.5 then
        break;
    end
    autores ← removeTopAuthor(A);
    tf++;
end
return tf;
```

Fonte: (AVELINO *et al.*, 2016)

valor 0, logo em seguida tem-se um laço de repetição que se itera sobre os arquivos do sistema, nele a cada ciclo é verificado a sua cobertura $(getCobertura(S,A))$, este que verifica a quantidade de autores presentes pela quantidade de arquivos no repositório. Caso o valor da cobertura seja inferior a 0.5 é encerrando o algoritmo retornando o valor atual do Truck Factor, caso contrário é feita a remoção do principal autor $(removeTopAutor(A))$ e acrescentado em $+1$ ao valor do Truck Factor.

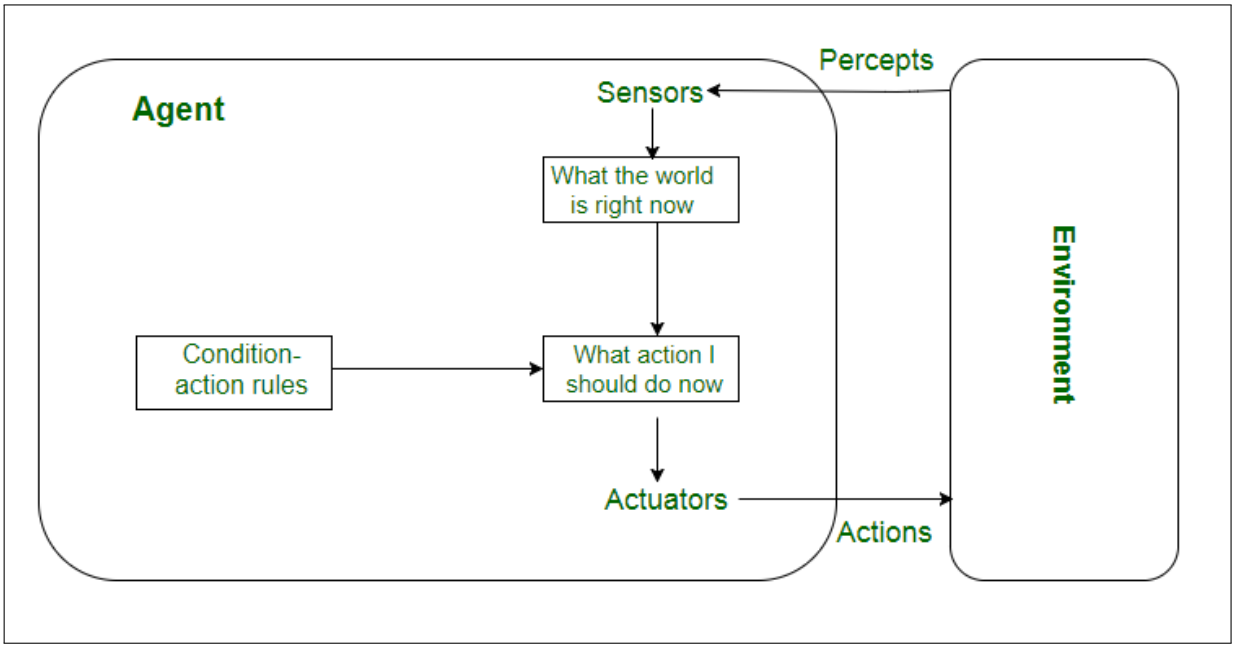
2.3 Sistemas Multiagentes

Para entender sobre Sistemas Multiagentes é necessário compreender o que são esses agentes. Segundo (WOOLDRIDGE, 2009), não existe uma definição exata na academia sobre sua denominação. No entanto, ainda de acordo com (WOOLDRIDGE, 2009) o agente é uma entidade computacional que está situado no ambiente, onde é capaz de realizar ações autônomas a fim de atingir seus objetivos.

Na Figura 2, é possível compreender como o agente interage com o ambiente. O agente percebe o ambiente através de seus sensores e realiza suas ações no mesmo por meio dos atuadores. De acordo com cada percepção obtida do ambiente o agente deve tomar uma determinada ação, tal ação tem como base cumprir seu objetivo individual.

Em muitos cenários, um único agente não é suficiente para resolver determinada problemática, logo torna-se necessário a atuação de mais de um agente no ambiente, a fim de solucionar tal problemática, nesse contexto multiagentes têm grande emprego. Multiagentes são uma extensão do uso de agentes inteligentes, onde estes são autônomos e atuam no ambiente

Figura 2 – Interação entre o Agente e o Ambiente



Fonte: (BANSALL, 2020)

para atingir seus objetivos. A cooperação, competição, compartilhamento de conhecimento e coordenação são exemplos de meios pelos quais os agentes podem realizar para atingir seus estados desejados(BALAJI; SRINIVASAN, 2010).

Os agentes possuem características, onde é possível classificá-los de acordo com as suas capacidades. O primeiro e mais simples é o agente reativo, nele os agentes têm apenas a função de reagir para determinadas percepções; o agente proativo demonstra seu comportamento direcionado aos seus objetivos; e a habilidade social é a capacidade de um agente interagir com outros agentes a fim de satisfazer seus objetivos (WOOLDRIDGE, 2009).

Todo agente deve estar presente no ambiente, através das suas percepções executa ações a fim de atingir seu objetivo. O ambiente tem a função de manter suas propriedades para que os agentes possam perceber e agir.

2.3.1 Ambiente

O agente deve obrigatoriamente estar inserido em um determinado ambiente, conforme apresentado na Figura 2, existe um ciclo de percepção de ação do agente com o ambiente. Segundo (RUSSELL; NORVIG, 2002) existe uma vasta quantidade de ambientes, porém é possível encontrar características em comum, que são denominadas de dimensões. Logo, através das características do problema é possível definir suas dimensões. Abaixo são apresentadas as dimensões existentes, bem como também suas definições.

1. Completamente Observável versus Parcialmente Observável:
   - Esta característica diz respeito à percepção dos agentes em relação ao ambiente. Se os agentes têm acesso a todos os atributos relevantes do ambiente entende-se que o ambiente é completamente observável. Em situações em que a percepção pode ser alterada devido ruído na leitura ou até mesmo a ausência da informação entende-se que o ambiente é parcialmente observável. Uma terceira situação ocorre quando o agente não consegue perceber nenhuma informação sobre o ambiente, logo denomina-se de inobservável.
2. Agente único versus multiagente:
   - Nesta dimensão deve compreender a natureza do problema e se em tal cenário haverá mais de um agente a completar seus objetivos. Em cenários que somente um agente deve atuar a fim de chegar em uma solução dizemos que é de agente único. No entanto, se há pelo menos dois agentes é um ambiente multiagente. No cenário multiagente existe o mecanismo de cooperação e competição entre os agentes, podendo ser parcial ou completo. Por exemplo: agentes que estão atuando como jogadores de um mesmo time de futebol, estão de um lado cooperando com seus companheiros de equipe, de outro ponto de vista estão competindo com a equipe adversária, logo neste cenário é parcialmente cooperativo e parcialmente competitivo.
3. Determinístico versus estocástico:
   - Nesta dimensão quando o ambiente é denominado determinístico entende-se que o próximo estado é determinado pelo estado atual do ambiente e pela ação realizada pelo agente. Quando tal situação não existe é chamado de estocástico. Em situações reais entende-se que praticamente todos os ambientes são estocásticos, visto que existem vários agentes tomando ações divergentes no ambiente. Como exemplo temos a direção de um carro pelas ruas da cidade, embora o motorista tenha percepção do ambiente e do carro, ele não consegue saber se o pneu irá estourar ou até mesmo se o motor irá parar de funcionar.
4. Episódico versus Sequencial:
   - Na dimensão episódica entende-se que a decisão tomada pelo agente é atômica, ou seja cada percepção que é recebida pelo o ambiente o agente escolhe sua ação e em próximos episódios a decisão tomada não terá influência. Por outro lado, em ambientes sequenciais cada ação tomada no episódio atual sofreu influência de ações

Tabela 1 – Exemplo de ambientes com suas respectivas dimensões

Dimensões	**Palavras Cruzadas**	**Xadrez**	**Análise de Robô por Imagem**	**Direção de Carro**
Observável	Sim	Sim	Não	Não
Determinístico	Sim	Não	Não	Não
Episódico	Não	Não	Não	Não
Estático	Sim	Sim	Não	Não
Discreto	Sim	Sim	Não	Não
Agente único	Sim	Não	Sim	Não

Fonte: (RUSSELL; NORVIG, 2002)

realizadas em episódios anteriores. Como exemplo sequencial tem-se uma partida de Xadrez, cada ação tomada pelo jogador, terá influência nas suas futuras jogadas.

5. Estático versus dinâmico:
   - Nas situações em que o ambiente pode ser alterado no mesmo momento que o agente está agindo, podemos dizer que o ambiente é dinâmico para o agente, quando isso não ocorre denomina-se estático. Ambientes estáticos são mais simples para se controlar, pois o agente não precisa ficar observando continuamente o ambiente para verificar as mudanças das variáveis.
6. Discreto versus contínuo:
   - A diferença entre ambas diz respeito de como o tempo é tratado, e ainda às percepções e ações do agente. Em ambientes que existem um número máximo de estados finitos denomina-se como discreto, temos como exemplo uma partida de Xadrez, onde para cada jogador existirá uma quantidade máxima de estados atingíveis. Nesse mesmo exemplo o agente possui um conjunto discreto de percepções e ações. Por outro lado, um agente que trafega no trânsito é um problema de estado contínuo e tempo contínuo, visto que a velocidade e a posição dos carros no ambiente passam por um intervalo de valores contínuos e fazem isso suavemente ao longo do tempo.

A combinação mais complexa de dimensões para o agente é quando ele é parcialmente observável, estocástico, sequencial, dinâmico, contínuo e multiagente. Na Tabela 1, tem-se alguns exemplos de aplicações de agentes com a descrição de suas dimensões. Nesta situação tem-se a aplicação de direção de carros através de agentes como a combinação de dimensões do ambiente mais complexo.

2.3.2 Comunicação

A interação entre agentes é a base do estudo de sistemas multiagentes, uma vez que surge a presença de agentes no ambiente, torna-se necessário a comunicação para que exista cooperação e coordenação. A passagem de mensagem é um método bastante utilizado entre agentes, conforme exemplificado a seguir: o agente A necessita saber alguma informação do ambiente ou até mesmo se certificar de uma determinada situação é possível que o mesmo envie uma mensagem diretamente ao Agente B, com sua determinada requisição.

Agent Communication Language (ACL) é uma abstração para troca de informação e conhecimento entre os agentes. Para que a comunicação seja efetiva entre os agentes é necessário uma ontologia, por meio dela são especificados os significados dos conjuntos de termos do ambiente, para que todos os agentes possam obter o mesmo entendimento.

2.3.3 Organização

A existência de uma coordenação de agentes torna-se necessária à medida que a quantidade de agentes dentro do ambiente é ampliada, logo é necessário uma estrutura para assegurar que as regras e normas sejam respeitadas. Por meio das organizações são definidos relacionamentos, papéis, níveis e normas para a comunidade de agentes. Na literatura são propostos modelos organizacionais, onde se define a forma de como essas organizações devem atuar no ambiente e se relacionar com outros agentes. Cada abordagem possui características que devem se enquadrar em determinados contextos.

2.3.4 Modelo Baseado em Agente de Simulação

Agent Based Modeling (ABM) são agentes computacionais que interagem de forma autônoma e buscam através de comportamentos e ações simular um determinado ambiente a fim de analisar e resolver problemas. Muitas plataformas de simulação são baseadas em bibliotecas ou ferramentas que descrevem um paradigma e conceitos próprios. Este modelo teve grande adesão em pesquisa dos mais variados campos de estudo, devido sua fácil aplicação em problemas que outros modelos computacionais não conseguem resolver (RAILSBACK *et al.*, 2006).

Como apresentado em (HONGQIAO *et al.*, 2009; CHAE *et al.*, 2015; BAIA, 2015), agentes baseados em simulação são largamente utilizados em pesquisas da área de Sistemas

Tabela 2 – Simuladores para Sistemas Multiagentes

Simulador	**Linguagem**	**Funcionalidades**
NetLogo	Logo	Ambiente completamente programável; Comunidade com modelos multiagentes prontos para uso; Ambiente 2D e 3D de simulação; Interface Gráfica da simulação; Executa Script de outras linguagens.
Repast	C++ e Java	Herda de bibliotecas de C++ e Java; Possui interface gráfica; Completamente orientada a objetos;
Spade	Python	Multiagentes baseado em comunicação XMPP; Agente baseado em comportamentos; Interface gráfica Web.

Fonte: elaborada pelo autor

Multiagentes, visto que nelas podem ser representados de uma maneira simplificadas e artificial muitas variáveis que estão presentes em ambientes reais, com o benefício que o processo pode ser rapidamente reproduzido em diferentes de condições, assim permitindo uma análise mais precisa do problema estudado.

Existem diversas ferramentas que possuem foco em simulação de agentes, na Tabela 2, é possível analisar um conjunto de ferramentas disponíveis com diferentes funcionalidades. Como apresentado, o Netlogo é uma ferramenta simples, porém poderosa que permite a programação e modelagem de ambientes multiagentes, adequada para o simulação de ambientes complexos e que simulam fenômenos naturais e sociais (NETLOGO, 2022). O Netlogo permite de maneira simples a utilização de interface gráfica dos agentes no ambiente.

O Repast é uma ferramenta open source e gratuita que permite a modelagem de ambientes multiagentes para simulação, onde oferece aos seus usuários uma rica variedade de funcionalidades (REPAST, 2020), a ferramenta é completamente focada em oferecer multi-thread aos agentes, também é possível visualizar cenários de simulação 2D e 3D.

O Spade é uma biblioteca python que permite implementação de multiagentes, ele se baseia na modelagem de agentes através de comportamentos, no Spade é possível trabalhar com multi-thread, tem suporte ao FIPA, onde permite comunicação entre os agentes (SPADE, 2020). Entretanto, não existe ambiente gráfico para a simulação, nesse caso seria necessário a implementação através de uma ferramenta ou biblioteca externa. No entanto, como o python possui uma variedade de bibliotecas gráficas é possível adicionar ao projeto e implementar tal ambiente gráfico.

2.4 Search Based Software Engineering

A Engenharia de Software (ES) surge da necessidade de apoiar a construção de software. Ela propõe a realização de métodos e processos que devem ser executados para uma boa gestão do desenvolvimento e manutenção de sistemas (SOMMERVILLE, 2010).

Dentro dos processos de desenvolvimento de software surgem problemas que são difíceis de se solucionar. Essas questões envolvem encontrar soluções adequadas entre objetivos concorrentes e potencialmente conflitantes (HARMAN *et al.*, 2009). Outra situação são problemas que resultam em encontrar uma solução em um grande espaço de busca ou até mesmo possuem solução desconhecida.

A SBSE busca solucionar problemas da Engenharia de Software associados aos problemas de otimização da área de Search-Based Optimization (SBO) (HARMAN *et al.*, 2012). Ela é utilizada em diversas áreas da ES como: Engenharia de Requisitos, Testes, Depuração, Bugs, Gerenciamento de Projetos, Alocação de Recursos e etc.

O termo SBSE foi abordado inicialmente por (HARMAN; JONES, 2001), onde propõe a utilização de técnicas de busca para a resolução de problemas da Engenharia de Software. Porém tal propósito já havia sido abordado por (Miller; Spooner, 1976), onde em sua pesquisa buscou solucionar um problema de geração de testes automatizados com técnicas de otimização.

De acordo com (HARMAN *et al.*, 2009), as questões abordadas na ES são problemas que muitas das vezes podem ser abordados por uma solução baseada em otimização. Nota-se que em muitas vezes o Engenheiro de Software do projeto pode se questionar sobre algumas necessidades da gestão da equipe como:

1. Qual o menor conjunto de casos de testes que devem ser realizados para garantir a qualidade do software?
2. Qual a melhor alocação de recursos para esse projeto?
3. Qual a melhor sequência para a refatoração deste código?

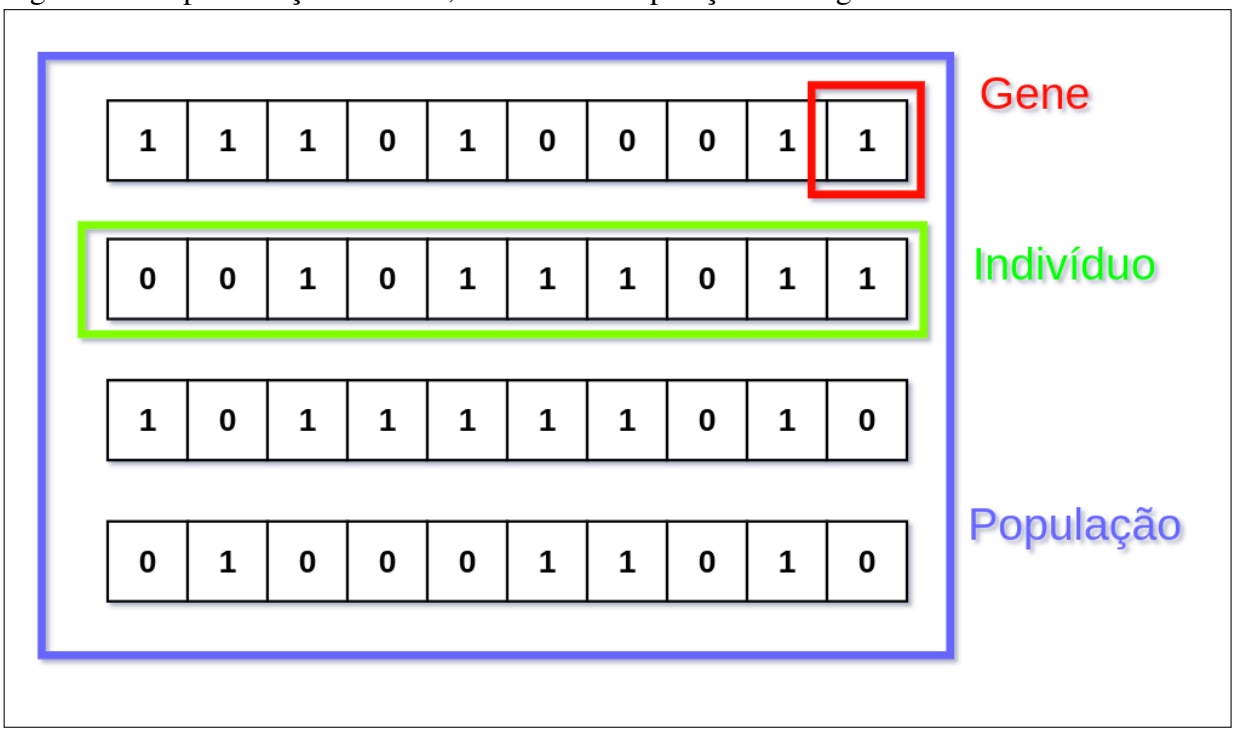
Essas questões acima são essencialmente problemas de otimização. Portanto, entende-se que SBSE tem mostrado um grande potencial para encontrar soluções em ES envolvendo questões de minimização ou maximização de um determinado problema.

Dentre as técnicas abordadas em SBSE estão os algoritmos genéticos, essa classe de algoritmos são baseadas na teoria desenvolvida da Evolução por Charles Darwin (DARWIN, 1968), é um algoritmo probabilístico baseado no princípio de sobrevivência dos mais aptos e na

reprodução usando princípios de seleção, na computação são geralmente utilizados em problemas otimização, pois fornecem uma soluções ótimas em tempo adequado (LAMBORA *et al.*, 2019).

Algoritmos Genéticos buscam encontrar uma solução considerada ótima para um determinado problema, seu ambiente é composto de uma população com indivíduos de características distintas, cada indivíduo é uma possível solução para o problema, ao longo do tempo somente os melhores indivíduos dessa população devem permanecer se reproduzindo e compartilhando seus genes. Cada indivíduo é avaliado, e de acordo com sua aptidão à solução terá maiores chances de ser selecionado para a reprodução. Somente os melhores serão reproduzidos para assim gerar uma população mais evoluída e cada vez mais próxima da solução. A ideia é que após um determinado tempo exista apenas um conjunto de indivíduos que se aproximem de uma solução considerada ótima.

Figura 3 – Representação de Gene, Indivíduo e População em Algoritmos Genéticos



Fonte: elaborada pelo autor

Na Figura 3, temos a representação do Gene, Indivíduo e da População. O indivíduo é caracterizado por um conjunto de genes, este que armazena as informações que estão relacionadas com a solução do problema, podendo ter diferentes maneiras de se modelar, no entanto, é comum implementação através de vetores. E por fim, a população caracteriza este conjunto de indivíduos.

Com a evolução dos algoritmos genéticos surgiram diferentes implementações, no entanto existem alguns processos que são comuns para o seu desenvolvimento. Conforme visto

Algoritmo 2: Implementação canônica do Algoritmo Genético

```
Result: Melhor indivíduo da população
Cria população inicial;
Calcula aptidão da população inicial;
while não atingir critério de parada do
    Seleciona os melhores indivíduos (Pais);
    Cruzamentos dos indivíduos selecionados;
    Mutação dos indivíduos gerados (Filhos);
    Calcula aptidão da população;
end
```

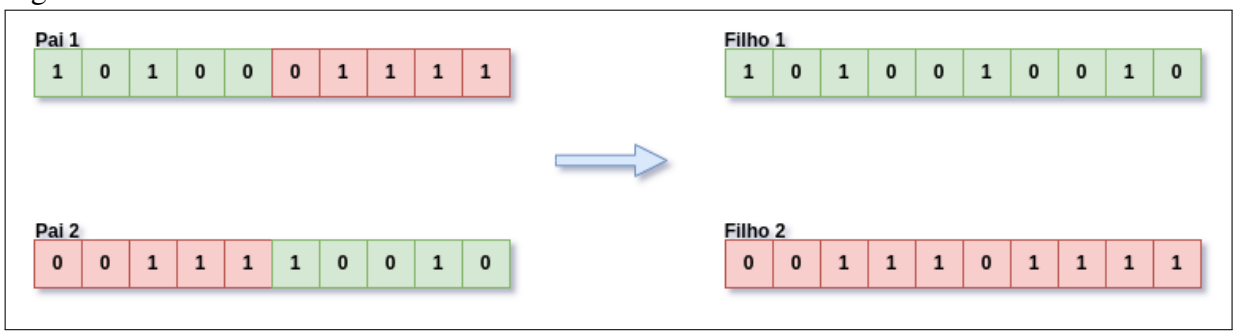
Fonte: elaborada pelo autor

no algoritmo 2, o processo se inicia através da geração da população inicial e o cálculo da aptidão da mesma, em sequência, inicia um processo repetido de seleção, cruzamento, mutação e aptidão dos indivíduos, isso deve se repetir até que um critério de parada seja atingido. Nos itens abaixo serão detalhados cada um dos processos realizados na implementação do algoritmo genético.

- **Seleção:** É considerada chave dentro do algoritmo, visto que dentre todos os indivíduos presentes na população, o algoritmo deverá selecionar somente alguns para seguirem para a próxima geração. Seu principal objetivo é garantir que os descendentes da próxima geração tenham uma aptidão ainda maior (MITCHELL, 1998). Uma seleção muito rigorosa faz com que somente indivíduos extremamente aptos assumam o controle da população, no entanto reduz a diversidade para mudanças e progressos. Por outro lado, uma seleção fraca, resultará em uma evolução muito lenta. Portanto, é ideal manter um equilíbrio no processo de seleção dos indivíduos.
- **Cruzamento:** É a escolha de dois indivíduos da população, que são denominados de pai, e realizam uma combinação de seus genes, a fim de gerar um novo indivíduo com parte da solução de cada pai. Podem existir diferentes maneiras de executar a combinação dos genes. Na figura 4, podemos observar uma situação onde acontece o cruzamento, temos denominado Pai 1 e Pai 2, e vemos a partir do seu cruzamento, surgem dois novos indivíduos, se observamos o Filho 1, recebeu os 5 primeiros genes do Pai 1 e os últimos 5 genes do Pai 2, de maneira contrária, mas seguindo a mesma lógica o Filho 2 foi gerado.
- **Mutação:** É um processo que realiza a cópia de um indivíduo da população anterior e modifica apenas alguns genes, este outro mecanismo que os algoritmos genéticos têm para garantir uma diversidade dentro de sua população. Como pode ser observado na Figura 5, vemos que após a cópia de um indivíduo, três genes do mesmo foram modificados.
- **Função de Avaliação ou Aptidão:** Momento onde cada indivíduo deve ser avaliado, tal

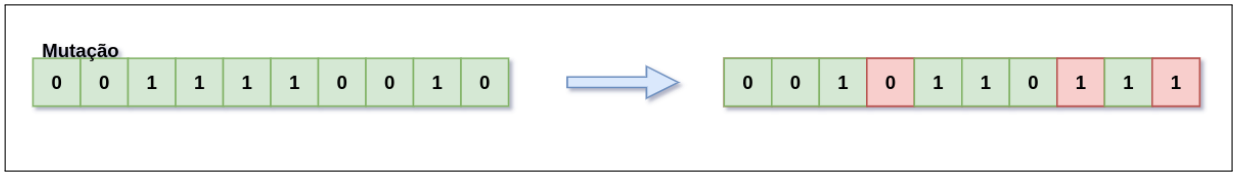
processo deve verificar o quão próximo de uma solução considerada ideal, essa avaliação está diretamente ligada ao problema em questão, visto que a avaliação do indivíduo terá impacto na seleção.

Figura 4 – Cruzamento de Indivíduos



Fonte: elaborada pelo autor

Figura 5 – Mutação de Indivíduos



Fonte: elaborada pelo autor

2.5 Considerações Finais

Neste capítulo, foram apresentados os principais conceitos que estão relacionados nesta pesquisa, que são: metodologia Ágil, Truck Factor, Sistemas Multiagentes, SBSE com algoritmos genéticos e modelo baseado em simulação de agentes.

Com relação à metodologia Scrum, foram apresentados seus principais aspectos, onde podemos observar que é possível adaptar a metodologia para diferentes ambientes. Portanto, é essencial entender que através dos métodos ágeis os processos podem ser adaptados para se adequar ao ambiente.

Em relação ao Truck Factor, foi realizado um levantamento de pesquisa que aborda tal temática, onde se propõe novas formas de se definir essa métrica ao projeto. A principal técnica apresentada foi a implementada por (AVELINO *et al.*, 2016), onde será utilizada para determinar o Truck Factor da simulação nesta proposta.

Em seguida, foi abordado o conceito de agente e sistemas multiagentes, descrevendo os diferentes tipos de agentes e ambientes. Em relação à sistemas multiagentes foi discutido

sobre a comunicação e organização desse tipo de sistema. Por fim, foi discutido sobre o modelo baseado em simulação de agentes, através da análise de ferramentas para modelagem de simulação multiagente, este que é fundamental para a construção desta presente proposta.

Por fim, foi discutido sobre a aplicação de algoritmo de otimização em problemas de engenharia de software. Em especial os conceitos e fundamentos elementares de algoritmos genéticos.

A fundamentação teórica apresentada neste capítulo fornece os fundamentos necessários para o desenvolvimento deste trabalho. O próximo capítulo apresentará a descrição da arquitetura proposta deste trabalho de dissertação.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos que se assemelham em algum aspecto com esta pesquisa, as quais servem como base ou comparativo para a mesma. Diante da busca por pesquisas semelhantes não foram encontrados artigos que buscam alocar tarefas em times de softwares com foco no *Truck Factor*, logo buscou-se levantar as pesquisas que utilizam técnicas e/ou contextos semelhantes a este, como: Sistemas Multiagentes, Search-Based Software Engineering, Alocação de Tarefas e *Truck Factor*.

Em (ANTONIOL *et al.*, 2004) é desenvolvida uma abordagem de planejamento para alocação de tarefas em equipes de software, que tem seu contexto direcionado para times de grande escala. O objetivo é encontrar uma solução ótima para distribuição das atividades entre as equipes de forma que o tempo de desenvolvimento do projeto seja minimizado. Visto que o problema é NP-completo na pesquisa foram utilizadas quatro abordagens de algoritmos, são eles: Algoritmo Genético, *Hill Climbing*, *Simulated Annealing* e também uma busca randômica. Para validação do estudo foram utilizados o histórico dos dados de uma empresa de software que trabalha no mercado financeiro. Além do foco em minimizar o tempo, outra questão importante para a pesquisa é encontrar o número ideal de pessoas no time. O autor conclui que de acordo com o cenário aplicado no estudo o *Hill Climbing* se mostrou mais eficiente, como também analisou que de acordo com o aumento do número de pessoas na equipe, se mostrou mais eficiente quando se atribui tarefas de longa duração.

Este estudo também possui o objetivo de distribuir tarefas dentre os membros da equipe, porém o principal parâmetro que será utilizado é o *Truck Factor*, parâmetro este que é fundamental para o bom desempenho e garantia de longa duração do projeto. Na pesquisa se utilizará algoritmos genéticos para se encontrar a solução ótima. A validação desta pesquisa será pela implementação através de sistemas multiagentes baseado em simulação, pois dessa maneira é possível simular diferentes ambientes da equipe, a fim garantir que o conhecimento esteja distribuído para seus membros.

Em (ROJAS; GIACHETTI, 2009) é apresentado um sistema de simulação baseado em agentes, onde é chamado de *Team Coordination Model* (Modelo de Coordenação de Times). A pesquisa tem foco na coordenação dos agentes no time como um processo influenciador da efetividade da equipe. O modelo foi desenvolvido para analisar a combinação da composição de equipes, sua característica organizacional, como também seus métodos de coordenação. No modelo cada agente possui uma função de tomada de decisão interna do agente, onde ele

avalia sua situação e realiza suas decisões baseada em um conjunto de regras. Visto que o experimento da pesquisa se deu através de simulação, o autor mostra que o processo simulado pode proporcionar uma boa oportunidade de avaliação em diversas configurações diferentes do sistema em um curto período de tempo.

Esta pesquisa baseia-se no modelo de simulação através de multiagentes, porém a configuração dos membros da equipe não influencia em seu resultado, visto que o objetivo principal é distribuir o conhecimento entre os membros ao invés de avaliar equipes diferentes e seu desempenho. No modelo apresentado por (ROJAS; GIACHETTI, 2009), a coordenação é descentralizada, dado que os próprios agentes podem entrar e sair de suas equipes a fim de otimizar o desempenho da equipe. Nesta pesquisa o papel é centralizado no agente do PO, uma vez que o processo de desenvolvimento do Scrum deve ser respeitado.

Em (FARHANGIAN *et al.*, 2016), a pesquisa tem o foco principal em criar um modelo de sistemas multiagentes, onde se possa oferecer suporte aos gerentes de projetos no momento em que são geradas novas equipes de software. A avaliação da pesquisa se baseou nas equipes que trabalham com *Python Enhancement Proposals* (PEP's). O modelo para formação de times de software se baseia na personalidade dos integrantes das equipes. Através das questões levantadas, a pesquisa visa mostrar como mensurar o desempenho da equipe baseado na combinação de personalidades. Para avaliar a personalidade das pessoas, a pesquisa utilizou-se de uma técnica do *Myers-Briggs Type Indicator (MBTI)*, nesta técnica a personalidade do indivíduo é enquadrada em 4 dimensões (*Extraversion/Introversion, Intuitive/Sensing, Thinking/Feeling, (Judging/ Perceiving*). Para verificar a performance do time, o autor verifica com o método de análise de personalidade todos os times do PEP's, a análise para o time é uma média e variância da personalidade dos membros. Em cada PEP tem uma tag que indica se houve sucesso ou falha no projeto. De posse dessas duas variáveis o autor então utilizou o algoritmo Apriori, que cria regras de associação para entre as personalidades com a performance do time.

Tanto em (FARHANGIAN *et al.*, 2016), como nesta proposta, tem-se o foco em atuar sobre gerenciamento de equipes de software, porém com contextos de aplicações diferentes, uma vez que neste estudo a personalidade do indivíduo não é levada em consideração, pois tal característica não influencia no *Truck Factor*. Nesta pesquisa o que se considera é um conjunto de habilidades do agente, assim cada membro tem um conjunto finito e distinto de habilidades, essas que são parte do critério para o agente PO para distribuir as tarefas. Já no estudo abordado por (FARHANGIAN *et al.*, 2016) seu foco principal se dá em formar equipe com distintas

personalidades a fim de otimizar o seu desempenho.

Em (EBADI *et al.*, 2008) a pesquisa engloba o contexto de cooperação em sistemas multiagentes, o autor desenvolveu um framework para SMA que é capaz de simular estratégias. O agente por padrão necessita de um parceiro para que a tarefa seja resolvida. A cada tarefa realizada o agente recebe uma recompensa, todos os agentes no ambiente são motivados a solucionar tarefas e assim intensificar suas recompensas. O resultado da pesquisa apresenta uma simulação, nela é vista as atitudes dos agentes em relação a seleção de parceiros para realização das tarefas. As tarefas surgem de tempos em tempos no ambiente, onde todos os agentes podem enxergá-la. Cada tarefa tem requisitos para que possa ser realizada, logo nenhum agente tem a capacidade total para realização da tarefa, visto que dessa maneira não seria necessário a formação de um time. Na pesquisa um time é definido como a junção de pelo menos dois agentes. Na pesquisa são apresentados 4 estratégias de seleção de parceiros: Melhor possível, Melhor disponível, Mais próximo disponível e Impaciente, onde na pesquisa os resultados apresentados utilizaram as 4 possíveis estratégias. Também foi implementada a atitude dos agentes, este mecanismo é responsável por sua tomada de decisão. Seus resultados apresentados indicam que em situações com maior quantidade de tempo, agentes que possuem a estratégia de Melhor Disponível e Impaciente, possuem melhor eficiência. No entanto, situações em que o tempo da tarefa é reduzido, a melhor estratégia de selecionar o agente mais próximo para realizar a tarefa se mostrou mais eficiente.

Como abordado por (EBADI *et al.*, 2008), é apresentada a necessidade da formação de times, uma vez que o agente nunca consegue realizar uma tarefa sozinho. Na presente pesquisa os agentes formam um time para realizar as tarefas, porém não há modificações destes times. Além disso, nesta pesquisa existe um agente centralizador que tem a função de distribuir essas tarefas. Por fim, as duas pesquisas abordam a ideia de resolução de tarefas, no entanto em contextos distintos. Para uma equipe de software não é compreensível que haja modificação constante na equipe, e quando houver uma troca de membros, o conhecimento acerca do projeto dentre a equipe garantirá que o projeto não perca desempenho na sua construção.

Em (YASIR *et al.*, 2018), a pesquisa entra no contexto de MG (Micro-Grids), que podem ser caracterizados como comunidades que produzem sua própria energia (Ex.: Energia solar). Nessas comunidades existe um problema que é a variação do recurso, em um momento há mais energia produzida do que consumida e outro em que o consumo é maior que a produção. Existem várias abordagens para se resolver essa problemática, o mesmo aborda uma ideia de

interconectar essas comunidades que produzem energia para que a troca de suprimento seja realizada entre as mesmas a fim de reduzir o impacto da irregularidade do recurso (Energia). Cada comunidade possui um agente que deve coordenar essa troca de recursos com as outras comunidades. Com a troca de recursos entre as comunidades surge o problema que caso uma cadeia de comunidades passem por dificuldades "desconforto" de seus recursos, ou seja, uma comunidades precisa de recurso, uma outra envia o recurso e com isso pode-se criar uma cadeia de necessidade. A comunidade é modelada como um agente inteligente, logo os agentes utilizam coalizão para minimizar o desconforto geral do recurso.

Embora em (YASIR *et al.*, 2018) o contexto de aplicação da pesquisa seja em um ambiente totalmente distinto, a problemática é semelhante, uma vez que ambos os problemas buscam distribuir uma quantidade limitada de recursos para agentes que a consomem. De forma semelhante, ao invés de minimizar o desconforto, como é denominado na pesquisa, o agente centralizador tem como objetivo minimizar o acúmulo de conhecimento dos agentes em relação ao projeto. Em ambas as pesquisas existe um papel de agente centralizador, já que é fundamental para garantir a distribuição dos recursos a fim de atingir seus objetivos.

Em (FARHANGIAN *et al.*, 2015), o autor também aborda um contexto de alocação de tarefas em equipes de software. Entretanto, inclui uma nova variável em relação às tarefas dos times, que são tarefas dinâmicas. A pesquisa utiliza uma simulação com sistemas multiagentes para a alocação das atividades da equipe, baseada na personalidade e habilidades da equipe, visando minimizar uma equipe super competente e outra com baixíssima competência. O pesquisador propõe uma ferramenta em que os gestores de projetos possam simular com um baixo custo a fim de receber insights sobre a estratégia de divisão de tarefas e pessoas com diferentes habilidades. Na pesquisa é levada em consideração a natureza dinâmica das atividades de um projeto, sendo que essa mudança (mudança de requisitos ou interdependência) afeta diretamente a efetividade da equipe. Quando o artigo aborda atividades dinâmicas, ele se refere às atividades que têm seus requisitos e outras variáveis modificadas. Sua contribuição se apresenta através da revisão aplicada na literatura e o desenvolvimento de um modelo para verificar a performance de um time. Além disso, é realizada uma análise entre a natureza dinâmica de uma tarefa com a estratégia de alocação de atividades pelos gestores. Os resultados foram apresentados através de um modelo simulado, onde utilizou-se o Netlogo para realizar a alocação de atividades.

No estudo proposto por (FARHANGIAN *et al.*, 2015), o autor mantém seu foco

em distribuir as tarefas entre seus membros, porém além do critério de habilidade é levado em consideração a sua personalidade, neste estudo a personalidade não deve ter influência sobre o conhecimento do agente sobre o projeto. Além disso, o autor também apresenta uma proposta para minimizar a distribuição de membros dentre os vários times, visando minimizar um time com alta competência e outro que não possui todas as habilidades disponíveis. Embora em times de software existam diversos times que trabalham em um mesmo projeto, entende-se que não há a necessidade de incluir mais equipes, visto que o foco não é na formação desses times e sim na distribuição do conhecimento dentre os seus membros.

A Tabela 3 demonstra semelhanças e diferenças entre os estudos relacionados. Como apresentado nesta pesquisa serão utilizadas as abordagens de SMA e SBSE, a validação será realizada através de um modelo simulado. É válido reforçar que, diante de um levantamento de pesquisas, não foi possível encontrar estudos que abordam alocação de tarefas no contexto de times de software com foco no *Truck Factor*, por meio dessas duas abordagens.

Tabela 3 – Comparação dos trabalhos relacionados divididos em critérios

Referência	**Simulação**	**Validação**	**Abordagem**	**Contexto**
(ANTONIOL *et al.*, 2004)	Não	Estudo empirico com dados reais	SBSE	Equipes de Software
(ROJAS; GIACHETTI, 2009)	Sim	Através de simulação	SMA	Otimização da composição de equipes
(FARHANGIAN *et al.*, 2016)	Não	Estudo empirico com dados reais	SMA	Performance de equipes de Software
(EBADI *et al.*, 2008)	Sim	Através de simulação	SMA	Contexto de Tarefas para Robôs autônomos
(YASIR *et al.*, 2018)	Sim	Através de simulação	SMA	Distribuição de Energia entre agentes
(FARHANGIAN *et al.*, 2015)	Sim	Através de simulação	SMA	Otimizar tarefas para times através da personalidade
Este Trabalho	Sim	Através de simulação	SMA/ SBSE	Otimizar tarefas para times de software com base no TF

Fonte: elaborada pelo autor

4 ABORDAGEM PROPOSTA

Neste capítulo será apresentada uma visão geral da abordagem, onde são definidos os agentes, ambiente, papéis, funções e como também serão as interações desses agentes. Também será detalhado a modelagem matemática do problema estudado. Em seguida, exemplifica-se através de um cenário de aplicação da abordagem. Por fim, é apresentado o processo de construção do ambiente multiagente através da simulação na plataforma NetLogo, como também um experimento do funcionamento do ambiente.

4.1 Motivação

A plataforma Netlogo foi utilizada para criar o ambiente multiagentes de simulação, em diferentes pesquisas recentes utilizaram esta ferramenta (BABIŠ; MAGULA, 2012), (ZHANG *et al.*, 2022) e (ZHOU *et al.*, 2022), pois dispõe vasta disponibilidade de recursos que necessitam ser utilizados na resolução do problema em questão. A simulação por meio do Netlogo torna o processo de validação da pesquisa mais viável, pois além de ser mais ágil do desenvolvimento do ambiente, na simulação é possível utilizar somente as variáveis necessárias para validar o processo.

Com o ambiente em funcionamento e respeitando os princípios necessários para executar o Scrum, buscou-se executar o experimento inicial no ambiente simulado, onde se analisou o comportamento dos agentes em relação à resolução das tarefas e ao conhecimento adquirido dos mesmos em relação ao projeto.

4.2 Visão geral da abordagem

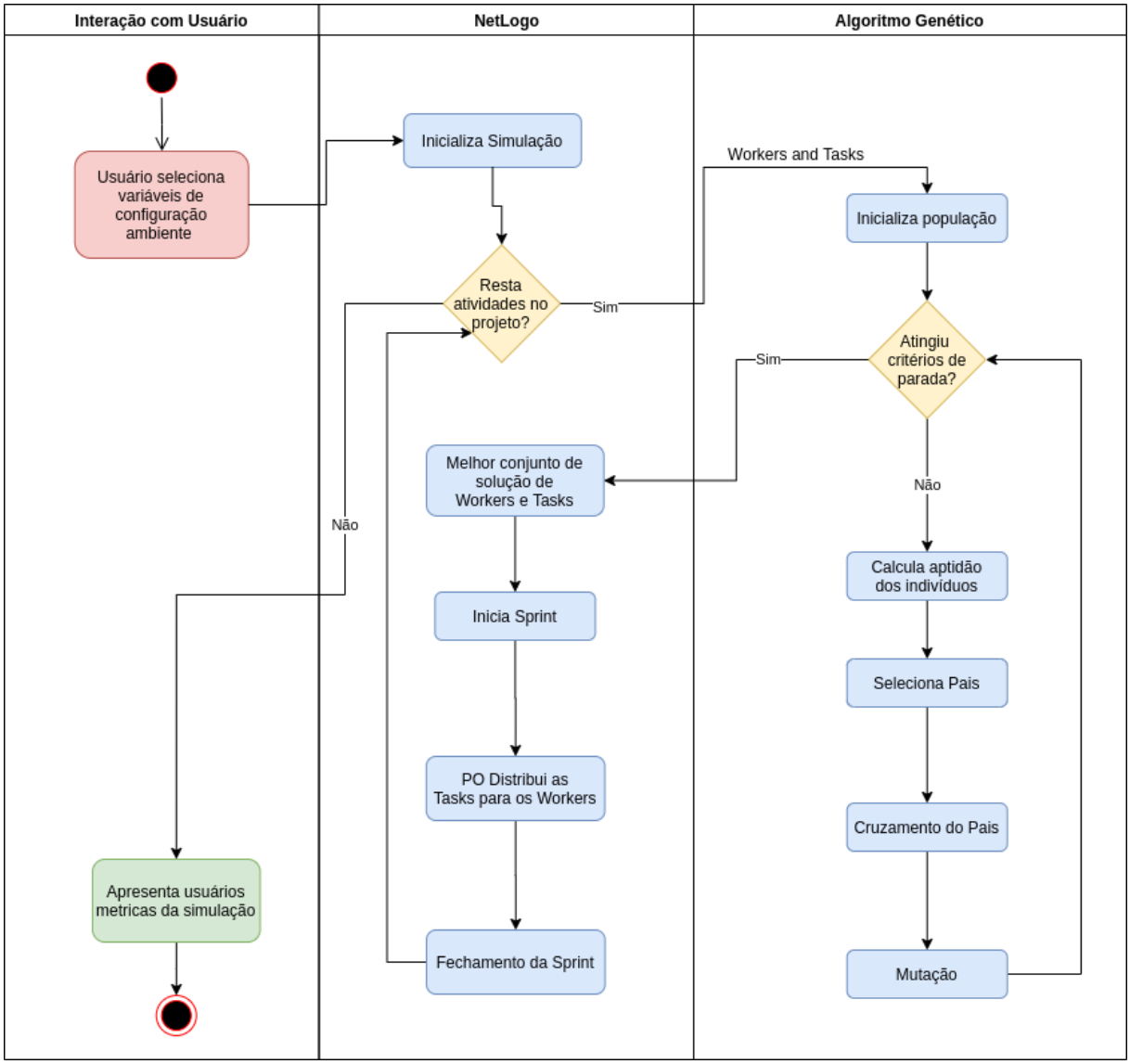
Na Figura 6 é apresentada a abordagem proposta através de algoritmos genéticos e multiagentes para a solução do problema. Como apresentado no diagrama de atividades, o modelo está dividido em 3 partes: Interação com usuário, Netlogo e Algoritmo Genético.

- Interação com usuário: Este é o momento onde o usuário deve definir as configurações para a simulação, pode ser definido variáveis como: quantidade de tarefas, trabalhadores disponíveis, habilidades disponíveis. Estão também disponíveis monitores para que o usuário tenha informações da simulação.
- Netlogo: É responsável pelo processo da simulação de fato. De acordo com as preferências do usuário definida no momento de interação a simulação deve ser iniciada. No Netlogo

deve ser executado todo processo relacionado com o scrum, onde contempla a abertura e fechamento da sprint. Também é realizado através do agente PO a distribuição das atividades, porém nesse momento o PO faz somente o link entre as tarefas e os trabalhadores disponíveis. O papel de otimização deve ser realizado pelo Algoritmo Genético.

– Algoritmo Genético: É responsável pelo processo de otimização. Neste momento o algoritmo terá como entrada a tarefas da sprint e os trabalhadores disponíveis, informações estas que serão enviadas pelo Netlogo ao python. O algoritmo genético deverá realizar todos os passos que são de gerar população, avaliação dos indivíduos, seleção, cruzamento e mutação, no fim do processo a melhor solução deverá ser retornada para o Netlogo.

Figura 6 – Diagrama de atividades da abordagem do sistema



Fonte: elaborada pelo autor

4.3 Descrição do problema

Neste problema temos $T_S = \{t_1, t_2, t_3, \ldots, t_n\}$, onde T representa o conjunto de tarefas que devem ser realizadas na sprint S, e $A_S = \{a_1, a_2, a_3, \ldots, a_m\}$ representa o conjunto dos agentes que nesse caso representam os recursos humanos disponíveis para a sprint, N e M representam o total de tarefa e recursos da sprint respectivamente. A solução será representada por um vetor S de tamanho N, logo temos $S = \{a_1, a_2, a_3 \ldots, a_n\}$, assim S_i armazena o agente responsável pela tarefa, e i a tarefa em questão.

Dentro de cada agente i, consideram-se os vetores $H_i = [h_1, h_2, h_3, \ldots, h_n]$ e $K_i = [k_1, k_2, k_3, \ldots, k_n]$, onde H representa seu conjunto de habilidades para resolução das tarefas e K representa o conjunto de conhecimentos que o agente possui sobre o projeto. Para a resolução da tarefa existe um vetor $L = [l1, l_2, l_3, \ldots, l_n]$ que representa o esforço necessário (nível necessário) para a conclusão da tarefa T_i, então o agente deve usar sua habilidade H_i, esta que deve representar a habilidade que o agente possui para concluir a tarefa. A tarefa é considerada finalizada quando $L_i = 0$.

A resolução de tarefa faz com que o agente possua maior conhecimento a respeito do projeto, logo K_i representa o conhecimento adquirido sobre o projeto em relação à habilidade H_i, portanto temos que $K_i = \sum_{i=1}^{s} L_i$, onde s representa a Sprint atual do projeto. Da mesma forma ao término de uma atividade realizada pelo agente é considerado uma matriz $R = ZxA_m$, onde R deve representar o repositório de código fonte do sistema e Z representa a quantidade de arquivos, este atributo é fundamental para se encontrar o Truck Factor do sistema. Portanto da mesma forma de K_i ao finalizar uma tarefa T_i, existe um atributo F_i que diz respeito aos arquivos necessários para a realização da tarefa, nesse temos que $R[F_i, A_i] = \sum_{i=1}^{s} Fi$.

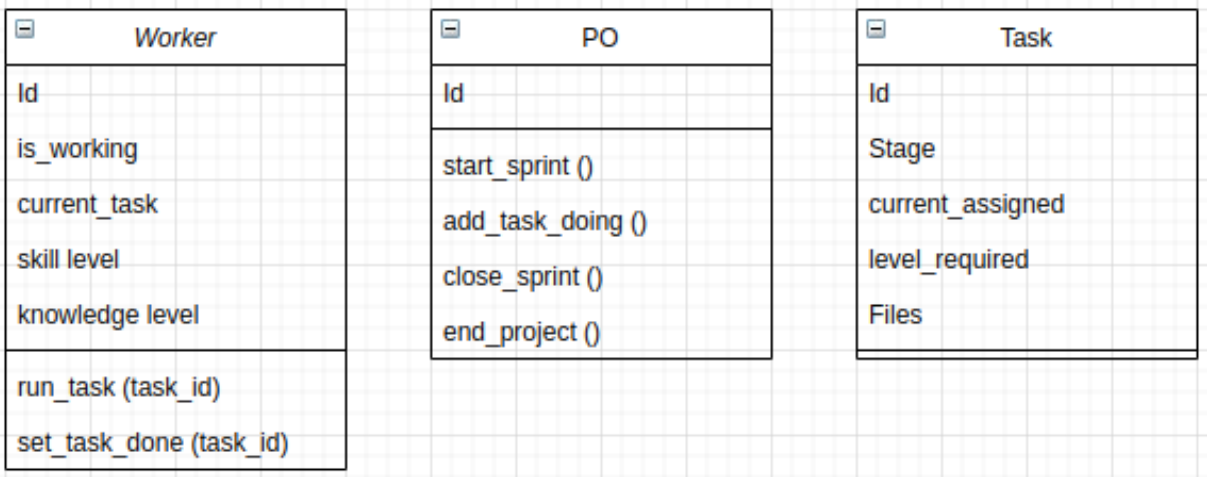
4.4 Construção do Ambiente

A primeira etapa necessária é a de criar o ambiente multiagente, este que deve incluir os parâmetros e variáveis presentes no contexto real de um time de software. O ambiente foi construído com base na metodologia Scrum, onde aborda seus papéis e métodos de gestão e desenvolvimento.

O ambiente possui 3 tipos de agentes, que são o agente PO, os agentes trabalhadores e as tarefas. Cada tipo de agente possui seus próprios atributos, essas variáveis têm a função de manter a percepção de cada agente sobre o ambiente.

Na Figura 7, é possível visualizar os atributos e procedimentos de cada agente. Obrigatoriamente o Netlogo atribui uma variável de identificação (Id) em todos os agentes para que possa fazer o controle da simulação.

Figura 7 – Atributos e procedimentos dos agentes



Fonte: elaborada pelo autor

No agente Worker, o atributo “is working“ deve manter o estado quando o agente está realizando alguma tarefa, “current task“ armazena o Link do agente com a tarefa atual, “skill level“ é um vetor que armazena o conjunto de habilidades que o agente possui e “knowledge level“ indica o nível de conhecimento sobre o projeto em cada aspecto de habilidade do agente. Além disso existem os procedimentos de “run task“ e “set task done“, que indicam a execução da tarefa e sua conclusão respectivamente.

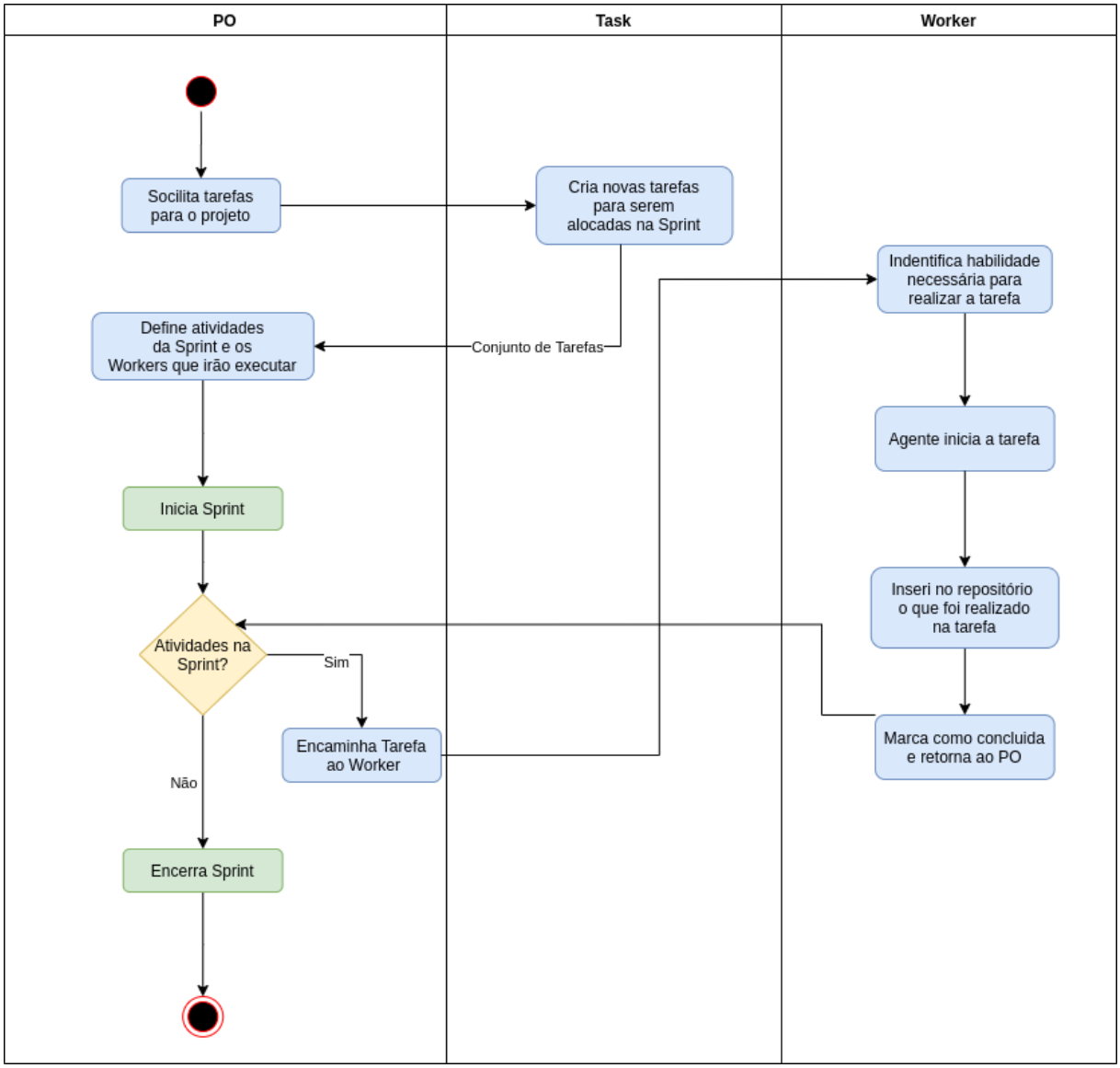
No agente PO a princípio se tem atributos de estado, porém sua principal função é de executar procedimentos, esses são de iniciar e finalizar a sprint, através das funções “start sprint“ e “close sprint“ respectivamente. Na função “start sprint“ as tarefas são retiradas do Backlog e incluídas na sprint. No procedimento de “add task doing“ é o momento em que o agente deve associar uma tarefa para o agente worker, esse é o principal método desenvolvido pelo PO, visto que nesse momento que o mesmo deverá levar em conta a alocação das tarefas de acordo com a distribuição do conhecimento. Por fim também é sua função a de finalizar o projeto por meio da função “end project“.

O agente de Task possui atributos que dizem respeito ao nível necessário que o agente worker precisa atingir para concluí-la, isso se dá por meio da variável “level required“. O atributo “stage“ mantém o estado da tarefa, é fundamental o PO perceber as tarefas que estão em execução, finalizadas e ainda não iniciadas. O atributo “current assigned“ armazena o link

para o agente que está desenvolvendo a tarefa e, por fim, “Files“ diz respeito aos arquivos no repositório do software que serão manipulados pela atividades.

Na figura 8, é apresentado o diagrama de atividade com as interações realizadas entres os agentes no ambiente. O fluxo inicia com o agente PO solicitando ao agente Task a criação de novas atividades para serem incluídas na sprint, nesse momento a Task deve gerar e atribuir valores dinâmicos aos seus atributos. Após essa etapa o PO já deve possuir atividades suficientes para iniciar o planejamento da sprint, que será de associar as Tasks aos Workers, esse passo de associação será o ponto principal para o PO minimizar os efeitos do TF. Por fim o PO com uma solução ótima em mãos já pode iniciar a sprint, esse último processo deve ser repetido até o momento que já não existam mais atividades pendentes, e então deverá fechar a sprint.

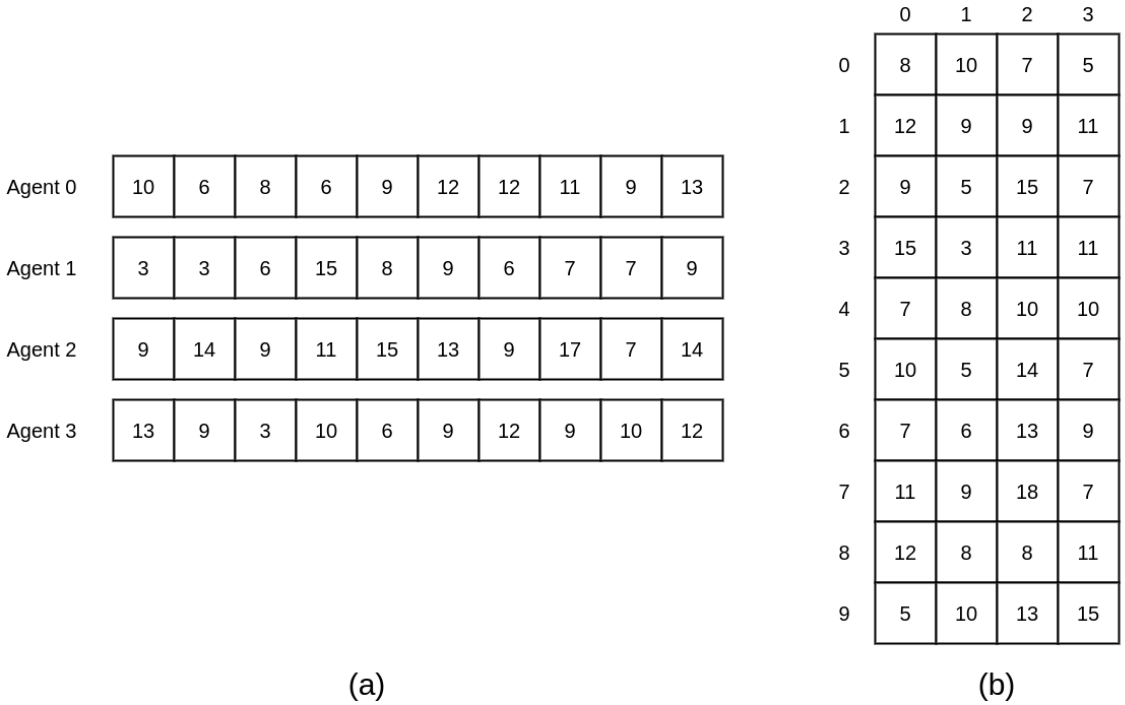
Figura 8 – Diagrama de Interação entre os Agentes



Fonte: elaborada pelo autor

Na figura 9(a), é apresentado o vetor de conhecimento de cada agente, na situação ilustrada existem 4 agentes, onde cada um deles possuem 10 habilidades. Este atributo deve armazenar todos os conhecimentos que o agente deve adquirir em diferentes aspectos a respeito do projeto. No início da simulação todos os agentes não possuem valores neste vetor, porém ao final de cada tarefa concluída pelo agente, o mesmo deve ter seu conhecimento acrescentado a respeito do projeto.

Figura 9 – Vetor de conhecimento e Matriz do repositório



Fonte: elaborada pelo autor

Cada posição no vetor de conhecimento representa a pontuação em uma determinada habilidade que o agente possui, na situação ilustrada o $agente1$ possui valor 15 na habilidade 3 do projeto, porém já na habilidade 0 e 1 têm apenas 3, quando se analisa apenas essas habilidades entende-se que há desequilíbrio no vetor de conhecimento do agente sobre o projeto.

A variância desse vetor de conhecimento é um aspecto onde demonstra problemas de concentração de conhecimento em determinadas partes do projeto, a situação ideal se apresenta quando a variância desse vetor de todos os agentes são minimizados, indicando que os mesmos possuem conhecimento distribuído em todas as partes do projeto.

Na figura 9(b), é apresentado a matriz que representa o repositório (“repository“), este que é definido como uma variável global do sistema. Este atributo é fundamental para se encontrar o Truck Factor do projeto, uma vez que nele são armazenados o mapeamento de

arquivos no sistema bem como o percentual de modificação de cada agente. As colunas devem representar os agentes presentes no ambiente, e cada linha deve representar arquivos do projeto. Logo os valores inseridos na matriz devem representar as contribuições dos agentes nos arquivos do projeto.

Em cada tarefa realizada pela equipe, os artefatos da atividade são inseridos no atributo “repository“. Através desse atributo é possível identificar a contribuição de cada agente no projeto, a ideia é simular um repositório de código fonte. Tal atributo é fundamental nesta proposta, visto que no algoritmo desenvolvido por (AVELINO *et al.*, 2016) (algoritmo 1) necessita deste parâmetro para encontrar o Truck Factor.

O código fonte implementado em python para cálculo do Truck Factor e do Algoritmo Genético encontram-se no apêndice B. O repositório completo, pode ser acessado através do link: https://github.com/caetanovns/resource-allocation-simulation. Na página principal, contém instruções de como executar o ambiente local e realizar os experimentos.

4.5 Considerações Finais

O presente capítulo teve como objetivo apresentar uma visão geral da abordagem proposta, onde foi descrito como os distintos componentes do sistemas irão de integrar para solucionar o problema, onde de maneira superficial o sistema se dividiu em 3 partes que são interface com usuário, Netlogo e algoritmos genéticos. O módulo de interface tem o papel de apresentar e interagir com o usuário durante o processo de simulação, como também ao fim da mesma apresentar o relatório com os resultados encontrados. O módulo do Netlogo fica com a responsabilidade central de gerenciar a simulação, visto que é nesta ferramenta que se gerencia as variáveis do ambiente e dos agentes. Por fim, o módulo do algoritmo genético atua na busca da solução ótima de cada Sprint para distribuir as tarefas entre os membros da equipe a fim de minimizar os impactos do Truck Factor.

No próximo capítulo serão apresentados os resultados obtidos através da construção do ambiente no Netlogo proposto neste capítulo, bem como se deu o processo de execução e coleta dos resultados na plataforma.

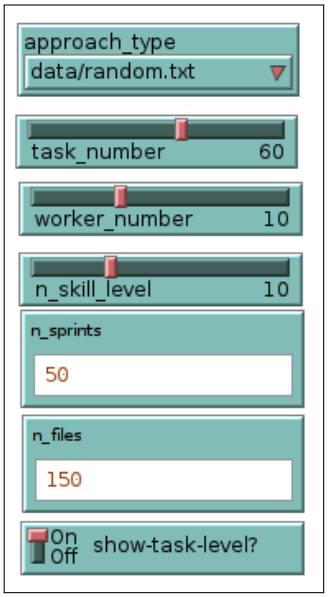
5 RESULTADOS

Neste capítulo, serão apresentados os resultados obtidos nesta pesquisa, desde o resultado da construção do ambiente de simulação dos agentes, como também a definição da solução do algoritmo genético, definição dos cenários que se realizaram os experimentos, até a execução dos experimentos com a coleta de métricas e análise sobre os resultados.

5.1 Simulação

No início da simulação, o usuário pode entrar com alguns parâmetros que podem ser configurados, essas variáveis podem ser observadas na Figura 10, a variável “task_number” definirá a quantidade de tarefa que surgirá a cada nova sprint, o parâmetro “worker_number” definirá a quantidade de pessoas que fazem parte do time; No “n_skill_level” tem o objetivo de definir as diferentes habilidade que os agentes podem ter para um determinado projeto, na “n_sprints” define de maneira geral o tamanho do projeto, quando tempo levará para ser finalizado, por fim em “n_files” define o tamanho do repositório de software, onde os agentes poderão contribuir durante a execução do projeto.

Figura 10 – Variáveis para configuração do ambiente

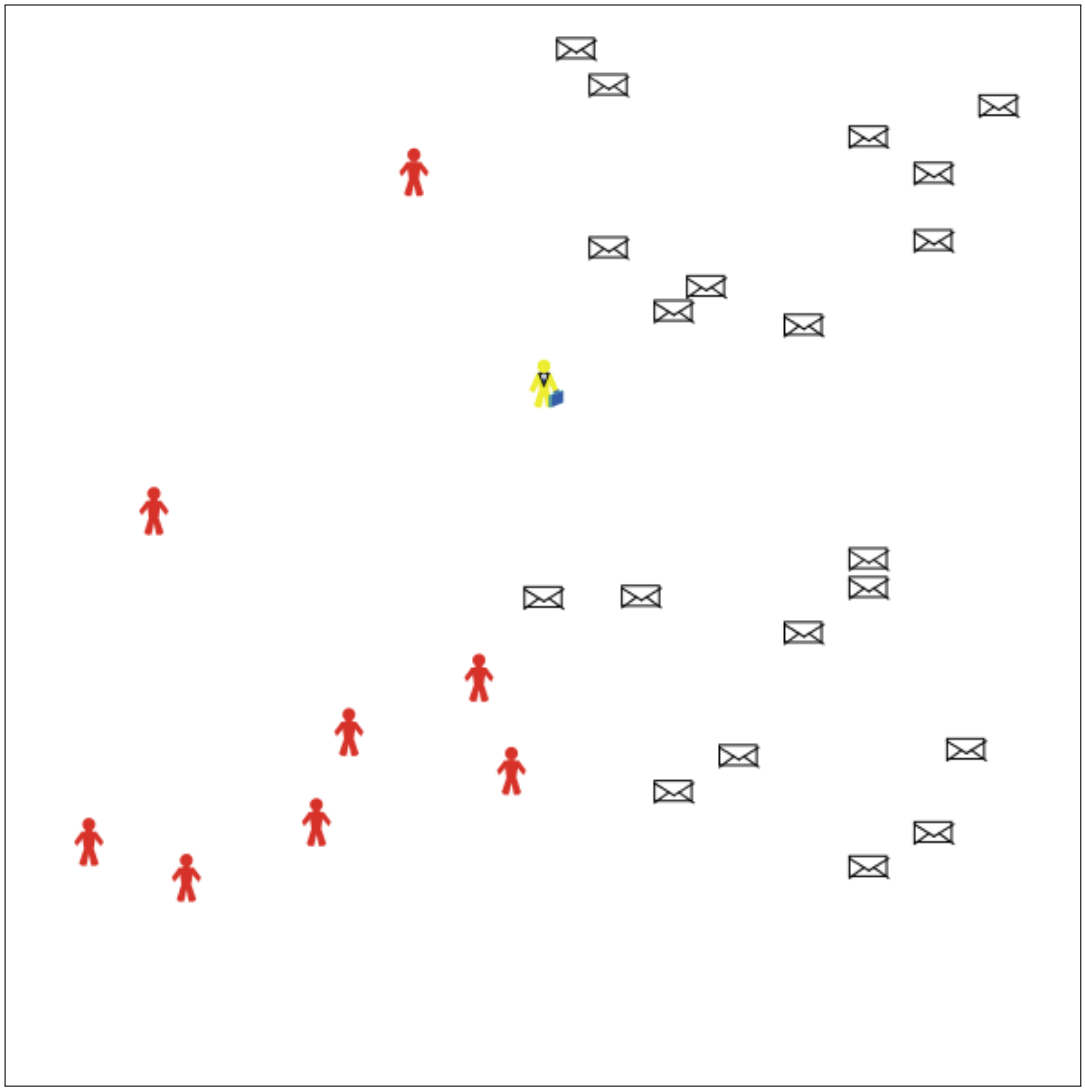


Fonte: elaborada pelo autor

Na Figura 11, é apresentado o momento do início da simulação. Os agentes quando estão em vermelho indicam que não estão realizando nenhuma atividade no momento, da mesma forma as tarefas que estão como o ícone de uma carta selada, indicam que ainda não

foram iniciadas por nenhum agente. Neste caso é possível notar que existem 8 (oito) agentes trabalhadores e nenhum está realizando nenhuma tarefa no momento e 20 (vinte) tarefas, e nenhuma ainda foi iniciada. O agente marcado em amarelo, é definido como agente único, pois tem o papel de distribuir as tarefas.

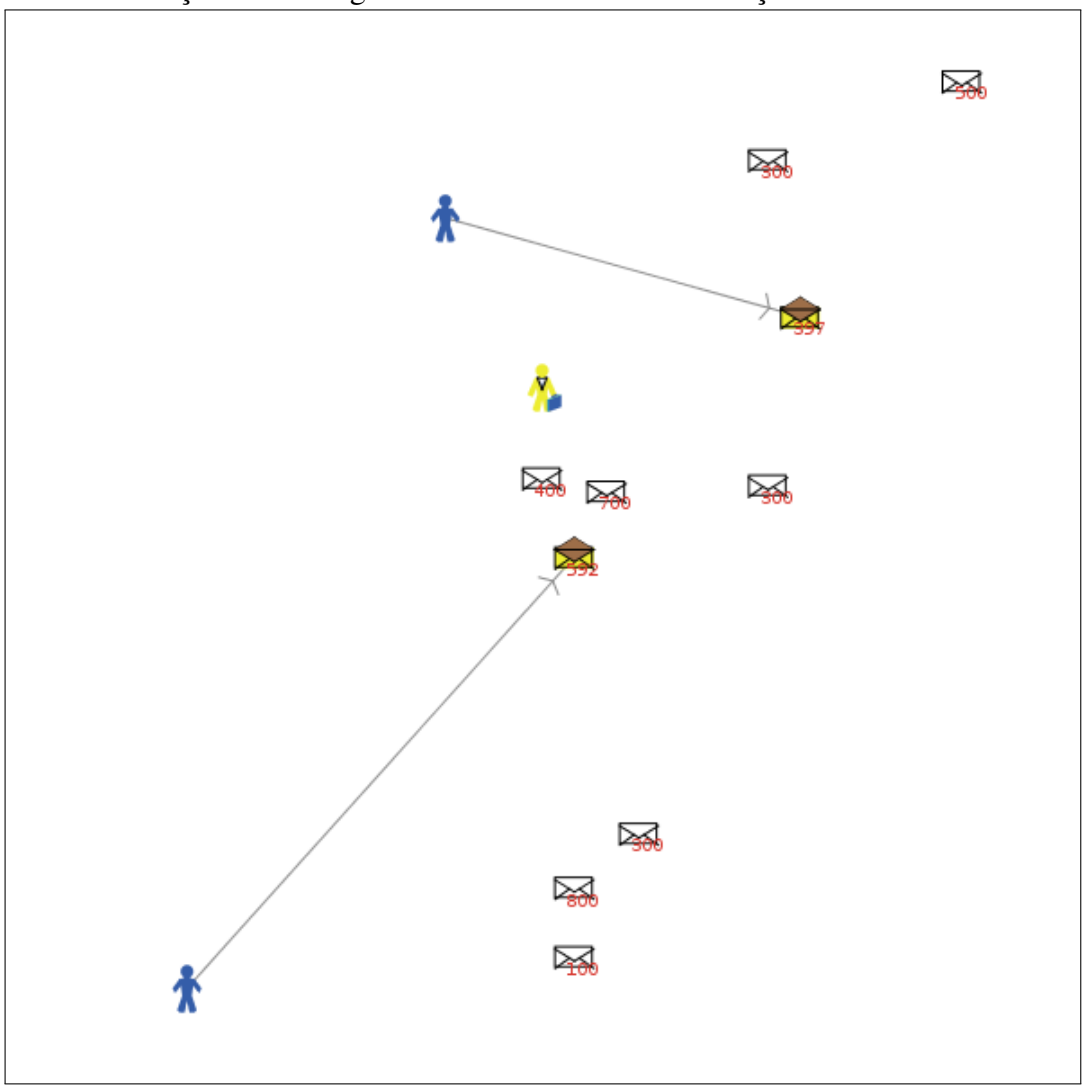
Figura 11 – Interação entre o Agente e o Ambiente pré simulação



Fonte: elaborada pelo autor

Na Figura 12, é apresentado o momento do início da sprint, no momento em que o agente estiver na cor azul indica que está associada em alguma tarefa, na interface a tarefa é indicada pelo arco que liga o agente à tarefa. Neste caso é possível notar que existem apenas 2 (dois) agentes trabalhadores e 10 (dez) tarefas, dentre elas 2 (duas) sendo realizadas no momento.

Figura 12 – Interação entre o Agente e o Ambiente com simulação iniciada



Fonte: elaborada pelo autor

5.2 Abordagem Proposta

A abordagem proposta nesta pesquisa é de aplicar Algoritmo Genético para Otimizar o parâmetro do Truck Factor. Adicionalmente, outras duas estratégias de alocação de tarefas foram desenvolvidas para serem utilizadas como comparação com nossa abordagem. Na abordagem denominada Otimista, as tarefas sempre são alocadas para o agente que possuir maior conhecimento a respeito do requisito necessário para finalizar a tarefa. A outra abordagem é a Random, nela não há nenhum princípio lógico para alocação, simplesmente as tarefas são distribuídas de maneira aleatória.

A implementação do Algoritmo Genético foi construído em Python, com a biblioteca DEAP, essa ferramenta oferece suporte para a construção do algoritmo, na Figura 13, temos um trecho principal da definição das funções do algoritmo, por meio da variável ‘toolbox’, podemos

definir as funções de criar os cromossomos, indivíduos, função de avaliação, população, mutação e seleção. As duas principais funções que devem ser implementadas são a definição do indivíduo, e sua função de avaliação.

Figura 13 – Base do algoritmo genético com DEAP

```
toolbox = base.Toolbox()

toolbox.register("chromosome", chromosome)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.chromosome, n=1)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)

best_solution = find_best_individual(toolbox)
```

Fonte: elaborada pelo autor

Através do diagrama, apresentado na Figura 14, podemos notar como foi definido o indivíduo, que é um candidato para ser solução do problema. O indivíduo é definido como um vetor, onde seu tamanho deverá corresponder a quantidade de tarefas da sprint, e em cada item do vetor deverá ser alocado um agente, que será o responsável por realizá-la, após o início da sprint. No exemplo apresentado na imagem, temos um vetor que representa 9 (nove) tarefas, e um conjunto de 5 agentes, os agentes podemos encontrá-los associados aos itens do vetor. O índice do vetor deverá representar a tarefa a que está associado, logo vemos por exemplo que o agente 3, está com 2 (duas) tarefas 2 e 3, o agente 1 (um) com as tarefas 0 e 5.

Figura 14 – Vetor de representação do indivíduo

0	1	2	3	4	5	6	7	8
1	0	3	3	4	1	2	4	2

Fonte: elaborada pelo autor

Alguns parâmetros do algoritmo genético precisam ser definidos, logo após sua implementação do algoritmo genético, foi necessário realizar experimentos a fim de encontrar parâmetros ideais. Para definir esses valores, foram feitos exaustivos testes no ambiente de simulação, com o objetivo do modelo fornecer uma solução adequada. Na Tabela 4 são apresentados seus parâmetros, para os cenários aplicados essa configuração se mostrou satisfatória. Os parâmetros de Validação Cruzada e de Mutação ficaram em 10%, isso significa dizer que apenas 10% dos indivíduos foram selecionados para fazer a validação cruzada, como também a mutação, o número máximo de gerações foi de 500 e com uma população inicial de 1000 indivíduos.

Tabela 4 – Parâmetros do Algoritmo Genético

Parâmetro	**Valor**
Validação Cruzada	0.1
Mutação	0.1
Gerações	500
População	1000

Fonte: elaborada pelo autor

Algoritmo 3: Algoritmo de função de aptidão do indivíduo

Input: individuo, tasks
Output: Aptidão do indivíduo
for $i \leftarrow 1$ *to* $len(individuo)$ **do**
    $worker = indiviuo[i]$;
    $task = tasks[i]$;
    $repository.commit(worker, task)$;
end
return $[variance(repository), truck\_factor(repository)]$;

Fonte: elaborada pelo autor

No algoritmo genético, todos os indivíduos são considerados um candidato para se tornarem uma solução definitiva do problema, com o objetivo de selecionar apenas os melhores candidatos ao longo das gerações, foi definida uma função para avaliar cada indivíduo, onde a partir de tal métrica será definido se a solução será selecionada ou não para a próxima geração está presente na próxima população de melhores indivíduos. De maneira geral, para a avaliação do indivíduo é levado em consideração às contribuições que o agente realizará no repositório de software, como também após as modificações do agente no repositório de software, qual o Truck Factor resultado daquele indivíduo.

A função de avaliação é apresentada no algoritmo 3 através de pseudocódigo, onde como entrada tem-se o indivíduo e um conjunto de tarefas da sprint, e saída a aptidão do indivíduo. O indivíduo contém as informações da tarefa, como também o agente que realizará a mesma, como já definido anteriormente na Figura 14. Dessa forma, são distribuídos às tarefas entre os agentes como proposto e em seguida, e realizando as mudanças no repositório de software. Após todo esse processo é analisada a variância dessa solução sob o repositório, como também o cálculo do Truck Factor, após todas as modificações serem feitas no repositório.

A métrica utilizada para comparar as diferentes abordagens implementadas foi a partir do resultado do Truck Factor, implementado no algoritmo 1. Esta métrica busca definir através de seu resultado a quantidade de pessoas em que o conhecimento sobre o projeto se encontra, para tal resultado utiliza como entrada o repositório de software para encontrar o valor

do Truck Factor. Uma vez que a abordagem dessa pesquisa é de minimizar os impactos causados pelo Truck Factor, logo a métrica do Truck Factor necessita ser maximizada.

Os dados gerados pela simulação foram obtidos do resultado do Truck Factor de cada sprint finalizada, visto que existe a necessidade de analisar seu comportamento ao longo do processo de desenvolvimento, portanto ao final de cada sprint finalizada o algoritmo do Truck Factor era executado e seu respectivo resultado armazenado em um arquivo.

5.2.1 Estudo Empírico

Neste experimento buscou-se mostrar a eficiência do uso da abordagem que utilizou-se a GA para a alocação de tarefas, onde maximiza a variável do Truck Factor. O experimento consistiu em elaborar alguns cenários distintos de projetos de software, que visam se assemelhar com uma configuração de projeto real. Na Tabela 5, são apresentados os cenários que foram realizados os experimentos. Em cada um dos cenários foi realizada a execução com a abordagem do Algoritmo Genético, Randômica e Otimista.

Tabela 5 – Configuração dos cenários de simulação

	Trabalhadores	**Tarefas**	**Sprints**	**Repositório**
Simulação 1	2	20	12	60
Simulação 2	6	30	30	100
Simulação 3	10	60	50	150
Simulação 4	15	85	50	150
Simulação 5	20	100	52	150
Simulação 6	25	100	52	300

Fonte: elaborada pelo autor

Ao final de cada caso de teste realizado, o Netlogo fornece um documento que contém os dados necessários para a análise das abordagens. O documento pode ser visualizado na Figura 15, nele contém o histórico completo do projeto no que diz respeito à métrica do Truck Factor. Nas colunas que iniciam com TF são armazenados os valores do Truck Factor por cada sprint, já nas colunas que se iniciam com *var_* é armazenada a variância de contribuição de cada agente em relação ao repositório.

5.3 Resultados da simulação

Nesta seção serão apresentados os resultados obtidos a partir da implementação da abordagem proposta. O principal resultado apresentado refere-se ao Truck Factor, visto que,

Figura 15 – Dados exportados do Netlogo ao fim da simulação

sprint	var_ga	var_random	var_best	knowledge_ga	knowledge_random	knowledge_best	tf_ga	tf_random	tf_best
1	15.04	16	18	510.11	506	2589	6	4	2
2	31.38	33	63	1110.97	1190	11353	9	7	2
3	42.94	49	118	1509.98	1789	25762	8	8	2
4	51.44	59	160	1883.59	2232	44371	10	8	2
5	57.44	68	212	2228.01	2753	66916	9	8	2
6	64.20	77	269	2943.98	3236	96017	11	9	2
7	73.21	88	338	3586.57	3933	136111	10	8	2
8	77.11	95	389	4023.25	4471	173606	10	8	2
9	85.17	106	444	4963.53	5420	232474	11	8	2
10	87.31	111	490	5402.93	5904	283090	10	9	2

Fonte: elaborada pelo autor

tem-se como principal objetivo da pesquisa a sua otimização.

Serão analisados posteriormente outros aspectos dos resultados, são eles: repositório de software, e conhecimento do agente. São utilizadas a fim de complementar o que se consegue perceber a partir da análise do Truck Factor. Em cada um dos diferentes aspectos analisados, são consideradas as 3 diferentes abordagens implementadas, que são: Otimista, Aleatório e a GA, onde a GA segue a implementação proposta nesta pesquisa.

A partir dos dados coletados, serão apresentadas análises através de estatística descritiva, onde os 3 diferentes aspectos dos resultados são analisados, posteriormente serão realizados testes estatísticos a respeito do Truck Factor, a fim de obter um resultado com um determinado grau de significância.

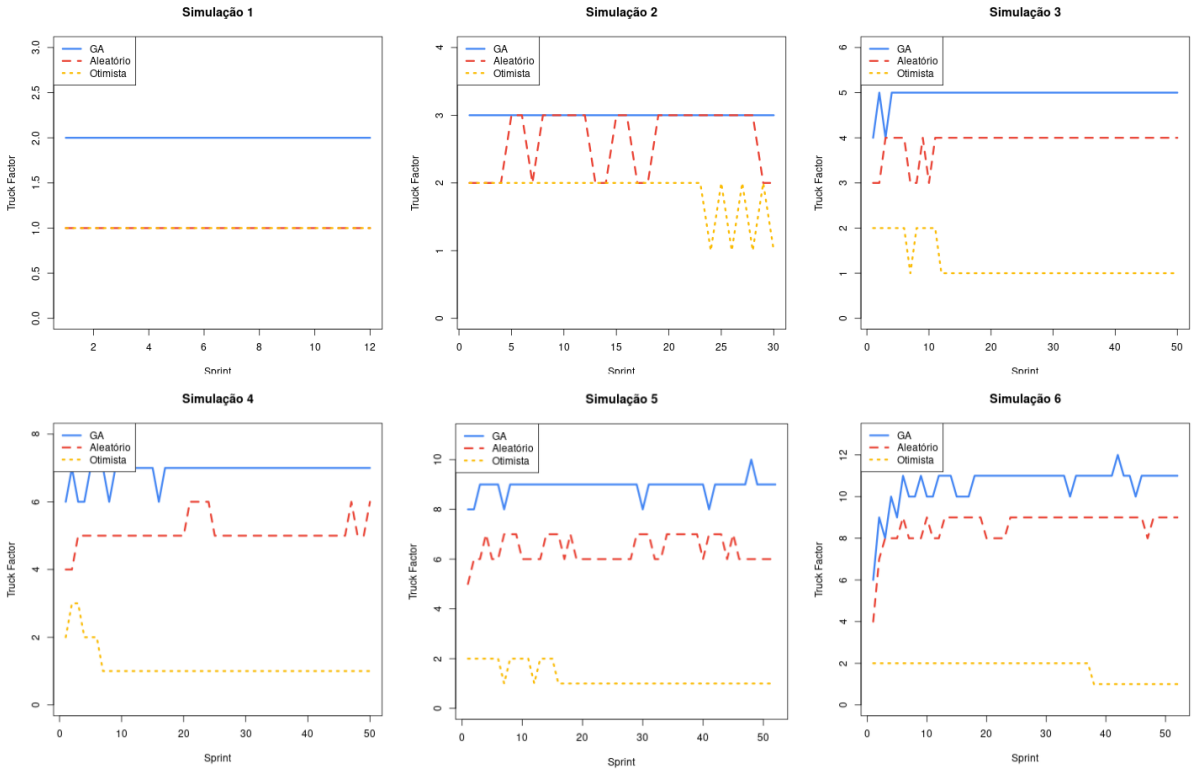
5.3.1 Análise do Truck Factor

Inicialmente foi realizada a análise estatística descritiva dos dados gerados a fim de analisá-los, os resultados da simulação foram analisados através da representação visual das informações, como também através de uma Tabela com os dados descritos.

Na Figura 16, estão representados através do gráfico de linhas os valores do Truck Factor ao longo da realização do projeto. A figura apresenta a união dos 5 cenários avaliados, conforme apresentado na Tabela 5. Em todos os gráficos são apresentados o desempenho de cada uma das abordagens desenvolvidas.

Analisando o gráfico é possível observar que em todos os cenários aplicados a abordagem Otimista obteve desempenho abaixo da abordagem Random e GA, isso é explicado pelo motivo por seu princípio de distribuição de tarefas, ele atribui as tarefas para o agente com maior desempenho técnico que a tarefa em questão necessita. Logo, é possível concluir que tal abordagem cria silos de conhecimentos entre os agentes do projeto e dessa forma o conhecimento a respeito do projeto fica concentrado somente em algumas pessoas.

Figura 16 – Dados do Histograma do Truck Factor



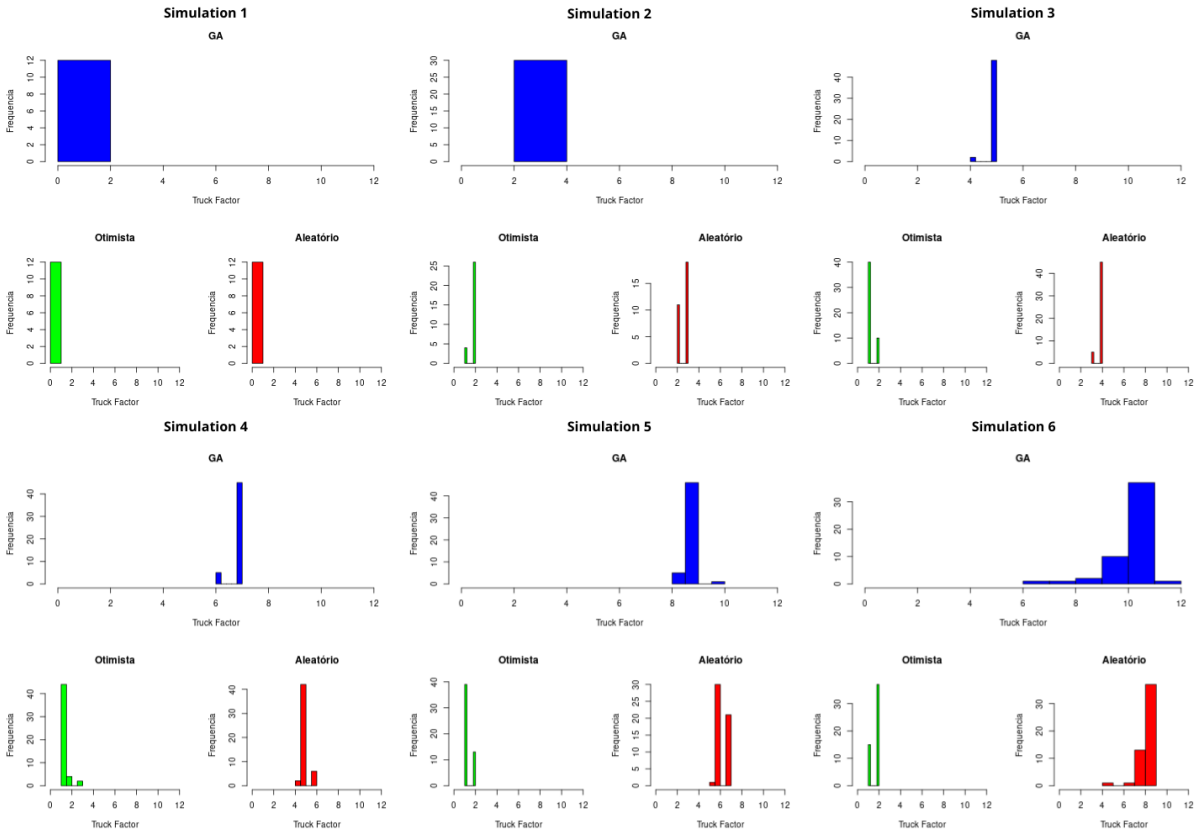
Fonte: elaborada pelo autor

As duas abordagens que obtiveram resultados mais próximos, foram a Random e GA. No entanto, é possível notar uma melhoria no desempenho da GA em comparação à Random, visto que em todos os cenários estudados os valores obtidos pelo Truck Factor ao final do projeto estão acima. Isso significa que em todos os cenários analisados, a GA conseguiu distribuir melhor o conhecimento entre os membros da equipe.

Quando se observa somente a métrica de avaliação do TF, a estratégia Random em alguns poucos momentos durante a execução do projeto se igualou a GA, como podemos observar na simulação 2, 3 e 6 da Figura 16. Essa situação ocorreu em cenários com projetos com baixa quantidade de pessoas (cenário 2), ou em momentos da fase inicial do projeto (cenário 3 e 6).

Na Tabela 6, são apresentados um resumo dos dados da comparação das abordagens analisadas em todos os cenários aplicados, também utilizando a métrica do Truck Factor. Por meio desses dados é possível visualizar de forma mais específica que nos cenários analisados a abordagem GA obteve desempenho superior em relação a Random e Otimista. É possível notar também que conforme o cenário se torna mais complexo, seja na quantidade de tarefas e de

Figura 17 – Gráfico de Barras do Truck Factor



Fonte: elaborada pelo autor

trabalhadores ou até mesmo no tamanho do repositório, a abordagem GA obtém um valor de Truck Factor superior a abordagem Random.

O histograma nos permite observar a frequência com o resultado do Truck Factor durante toda a duração do projeto, como pode ser observado na Figura 17, podemos notar que a frequência na abordagem “Otimista” se mantém praticamente constante, porém com um baixo valor de TF, o que entendemos ser insatisfatório. Na simulação 1, 4 e 5, nota-se que embora exista uma pequena variação, a maior parte da frequência se repete no menor valor do Truck Factor. Diferente do que se observa nos outros cenários (2, 3 e 6), onde embora ainda exista um TF insatisfatório a frequência é o máximo que essa abordagem consegue atingir.

Quando observamos na abordagem GA, podemos notar que em cenários menores (1 e 2) o valor se mantém constante, em cenários intermediários (3 e 4) existe uma pequena variação na frequência, porém maior parte dela está acima, o que é algo bom. Em cenários maiores (5 e 6) nota-se uma curva crescente na frequência no valor do TF até atingir seu valor máximo, posteriormente acaba ultrapassando este valor, porém não consegue manter e acaba

Tabela 6 – Estatística descritiva das simulações

Simulação	Abordagem	Média	Mediana	Moda	Desvio Padrão
Simulação 1	GA	2	2	2	0
	Aleatório	1	1	1	0
	Otimista	1	1	1	0
Simulação 2	GA	3	3	3	0
	Aleatório	2.63	3	3	0.4901
	Otimista	1.87	2	2	0.3457
Simulação 3	GA	4.96	5	5	0.1979
	Aleatório	3.9	4	4	0.3030
	Otimista	1.2	1	1	0.4041
Simulação 4	GA	6.9	7	7	0.3030
	Aleatório	5.08	5	5	0.3959
	Otimista	1.16	1	1	0.4041
Simulação 5	GA	8.92	9	9	0.3341
	Aleatório	6.38	6	6	0.5297
	Otimista	1.25	1	1	0.4372
Simulação 6	GA	10.60	11	11	0.9343
	Aleatório	8.62	9	9	0.8202
	Otimista	1.71	2	2	0.4575

Fonte: elaborada pelo autor

retornando ao valor anterior de maior frequência. Da perspectiva do Histograma, o Random de maneira geral obteve resultados semelhantes ao GA, porém notamos que no cenário 2 e 5 a frequência do TF do Random se mostrou inconstante, diferente da GA que se manteve estável e com TF superior.

5.3.2 Análise do Repositório de Software

Na seção 5.3.1 foram apresentados os resultados da simulação, onde foi realizada a análise do resultado sob a métrica do Truck Factor. Através desses resultados podemos observar que a abordagem GA obteve desempenho superior em relação a Random. No entanto, notamos também que embora o resultado do Truck Factor na maior parte do tempo seja superior, observamos que em alguns momentos a abordagem Random obtém resultados próximos. Como já apresentado anteriormente, na Figura 16 (Simulação 2), deixa bem evidente essa aproximação entre a GA e Random.

Em relação a abordagem Otimista, percebeu-se que em nenhum dos cenários foi satisfatória, obtendo um baixo Truck Factor, por esse motivo nesta seção será dado enfoque para a abordagem GA e Random. No entanto, para permitir uma visão completa sobre todos as diferentes abordagens, nesta seção está os resultados obtidos pela abordagem otimista continuam

Algoritmo 4: Função de avaliação do repositório do projeto

```
Input: Repositório do projeto
Output: Média da variância do repositório
variancia = []
for i ← 0 to len(repositorio) do
    variancia.add(variance(repositorio[i]));
end
return mean(variancia);
```

Fonte: elaborada pelo autor

sendo incluídos nos gráficos e tabelas.

A fim de investigar, as razões pela qual a abordagem Random em alguns momentos se aproximou dos resultados obtidos pela GA, e fornecer uma análise mais aprofundada dos resultados, analisaremos os mesmos dados, porém de uma nova perspectiva, utilizaremos a métrica da variável do repositório de software.

Essa métrica de avaliação do repositório, se dá através da variação da contribuição dos agentes. É uma medida mais sensível para encontrar diferenças sobre as abordagens analisadas no processo de distribuição de tarefas. Diferentemente da métrica do Truck Factor, que quanto maior melhor, a variância se mostra mais eficiente o quão menor seja essa variação, logo esse parâmetro deve ser minimizado. Os valores obtidos sobre a variância do repositório foram obtidos no mesmo momento em que fora realizada a simulação onde se coletava a métrica do Truck Factor, como pode ser observado na imagem 15, por meio das colunas de var_ga, var_random, var_best.

Conforme apresentado na seção 4.4 na Figura 9, o repositório possui uma estrutura de uma matriz, onde define em cada arquivo do repositório do software o quanto cada agente contribuiu, dessa forma a métrica aplicada é com base na variação dessa contribuição do agente. O processo de avaliação foi realizado ao final de cada sprint.

No algoritmo 4, podemos visualizar o pseudocódigo para a avaliação do repositório. Podemos notar que a única entrada é dada pela matriz que armazena os dados de cada agente em relação aos arquivos do projeto, e sua saída esperada é uma média da variância de cada arquivo do projeto. A implementação é realizada pela interação da matriz, onde a cada linha que representada o arquivo, a mesma calcula a variância e armazena em um vetor (variável "variancia"), ao final o mesmo realiza uma média simples do repositório e o retorna.

Na Figura 18, é apresentado o resultado em relação à métrica da variância do repositório. Analisando o gráfico, notamos que em todos os cenários analisados, mesmo com

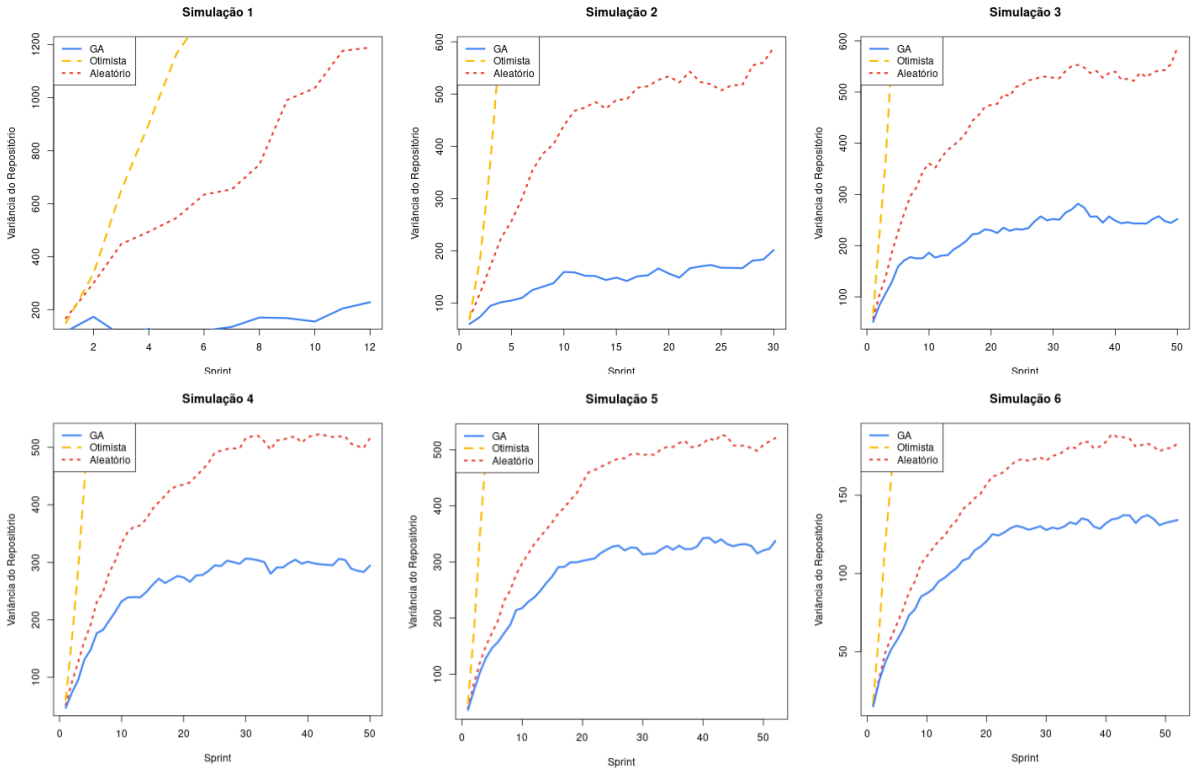
essa nova perspectiva, a abordagem Otimista se manteve como pior mecanismo de alocação das tarefas, pois sua curva cresce de maneira vertiginosa. Esse comportamento se manteve em todos os diferentes cenários observados.

Em relação às outras duas abordagens, podemos observar que embora exista sempre um crescimento do valor da variância, a GA, cresce de forma mais suave ao longo do projeto quando comparado com a Random, isso significa dizer que as contribuições realizadas pelos agentes no repositório de software são distribuídas de maneira equilibrada. Nos cenários 3, 4, 5 e 6 que são projetos com maior quantidade de pessoas, tempo de execução e tamanho do repositório, notamos que nas primeiras execuções, o valor da variância do repositório entre a GA e Random se aproximam, no entanto, ao longo do tempo acabam cada vez mais se distanciando. Por outro lado, nos cenários 1 e 2, observamos que desde o início a curva da GA se mantém distante da Random. Essas diferenças são causadas pelas características do projeto, quando temos cenários com menor quantidade de pessoas, tempo e repositório, a abordagem Random apresenta dificuldade na distribuição das tarefas, pois os recursos são restritos, por outro lado a GA nesse cenários se mostra eficiente, visto que consegue se adequar em ambientes com recursos limitados. Portanto, através do gráfico com a abordagem GA, podemos notar que a variação da contribuição, ou seja, o quanto cada agente contribui no repositório de software está distribuída de forma similar, onde mostramos que o conhecimento em relação ao repositório está distribuído entre os agentes.

Na Tabela 7, são apresentados com detalhes o resultado da variância do repositório de software das diferentes abordagens em todos os cenários. Na coluna de “última sprint“ é armazenado o valor da variância do repositório na última sprint realizada na simulação, isso significa dizer que é o resultado final do Truck Factor que o projeto obteve. Na coluna de “Diferença para GA (%)”, é armazenado o percentual de diferença das abordagens Random e Otimista para a GA, utilizamos este parâmetro para analisar com maior clareza a diferença entre as abordagens propostas.

Notamos que em cenários cada vez menores a GA se destaca a frente da Random, visto que a simulação 1 obteve seu melhor desempenho, com a diferença de 421.97%, como é possível observar na Tabela 7. Esses cenários de projetos menores, exigem mais do algoritmo em relação da distribuição das tarefas, visto que os recursos são extremamente limitados, tanto em relação ao tempo, quanto a quantidade de pessoas e tarefas. Seu desempenho mais baixo ocorreu na simulação 6 de 36.44%, onde é o cenário que existe maior quantidade de recursos

Figura 18 – Variância da variável do repositório de software



Fonte: elaborada pelo autor

disponíveis e tempo de projeto, no entanto esse resultado se mostra com um percentual de diferença satisfatório.

Na Tabela 8, buscamos complementar a análise já realizada através da análise da última sprint. Nela são apresentados o resumo dos dados da comparação das abordagens analisadas em todos os cenários aplicados, no entanto apresentamos com os valores de médias, mediana, moda e desvio padrão de todo o ciclo obtido no projeto. Podemos notar na tabela em as médias obtidas pela GA, quando comparadas a Random são inferiores, além disso outro aspecto importante é notar que além das médias, o desvio padrão se mantém inferior, isso nos mostra que se manteve mais constante durante todo o projeto, no aspecto da distribuição das tarefas. Embora como já demonstrado a otimista obteve um resultado pouco satisfatório, seus resultados foram incluídos a fim de proporcionar maiores detalhes.

Na Figura 19, temos um gráfico boxplot com dados extraídos da Tabela 8, por meio do boxplot temos uma análise visual da posição, dispersão e simetria dos dados. Através dessa nova forma de representação conseguimos perceber melhor a diferença entre as abordagens implementadas. Neste gráfico a abordagem otimista foi retirada por dois motivos: 1. Através

Tabela 7 – Análise resultado da última sprint do projeto

Simulação	Abordagem	Última Sprint	Diferença para a GA (%)
Simulação 1	GA	227.8	-
	Aleatório	1189.0	421.97%
	Otimista	1964.0	862.20%
Simulação 2	GA	201.6	-
	Aleatório	589.0	192.12%
	Otimista	2553.0	1266.18%
Simulação 3	GA	252.0	-
	Aleatório	587.0	132.95%
	Otimista	6313.0	2505.36%
Simulação 4	GA	294.2	-
	Aleatório	516.0	75.40%
	Otimista	4750.0	1614.66%
Simulação 5	GA	337.6	-
	Aleatório	521.0	54.32%
	Otimista	6470.0	1916.36%
Simulação 6	GA	134.1	-
	Aleatório	183.0	36.44%
	Otimista	1613.0	1202.65%

Fonte: elaborada pelo autor

Tabela 8 – Análise exploratória da variância do repositório

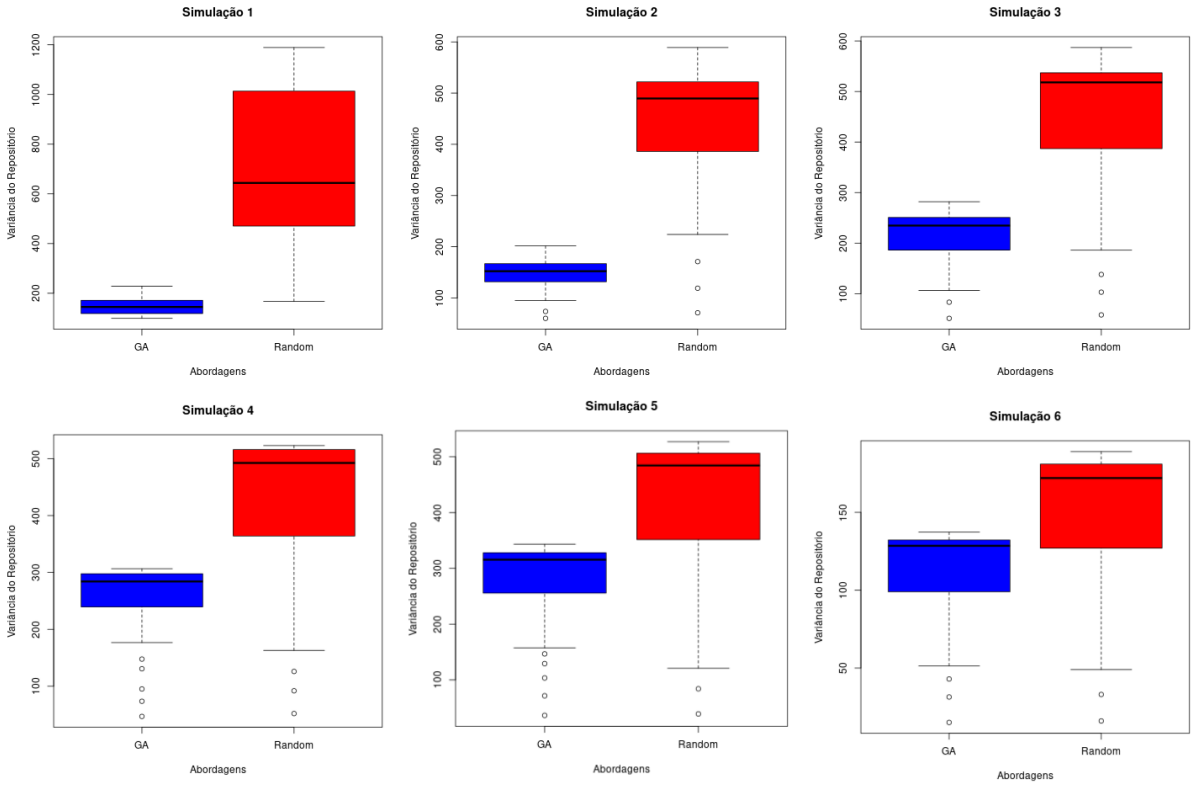
Simulação	Abordagem	Média	Mediana	Desvio Padrão
Simulação 1	GA	150	145	40
	Aleatório	698	644	337
	Otimista	1214	1389	588
Simulação 2	GA	145	152	33
	Aleatório	435	490	138
	Otimista	1746	1967	757
Simulação 3	GA	219	235	49
	Aleatório	448	518	130
	Otimista	3836	4173	1929
Simulação 4	GA	258	284	63
	Aleatório	422	493	127
	Otimista	2998	3417	1929
Simulação 5	GA	280	316	76
	Aleatório	2261	485	131
	Otimista	4107	4678	2097
Simulação 6	GA	112	129	30
	Aleatório	149	172	45
	Otimista	1063	1260	500

Fonte: elaborada pelo autor

da tabela já é notório o seu resultado e devidos seus valores estão em escalas muito maiores; 2. Inviabiliza a visualização das outras abordagens.

Entre a abordagem GA e Random, podemos perceber que em todos os cenários que a posição dos dados da GA são inferiores à Random, na maior parte dos cenários a mediana da GA se mantém inferior ao 1ª quartil da Random. Podemos observar também em relação a dispersão, os dados entre o 1ª e 3ª quartil da GA tem menor variação quando se compara com a Random, isso mostra que a variância na GA se manteve mais constante. Outro ponto que se nota a respeito da dispersão, é que ao longo do crescimento das variáveis do projeto (simulação 1 a 6), a dispersão entre os dados do 1ª e 3ª quartil se mantém crescente, no entanto ainda bem inferior a Random. Outro aspecto que se deve observar, são os outliers, estes que representam dados fora do padrão. Ao longo das simulações, observa-se a presença desses outliers, isso ocorre pelo fato de no início do projeto a variância inicial ainda não representar a maior parte dos dados, podemos observar isso de maneira intuitiva como já mostrado na figura 16, notamos que existe um crescimento acentuado no início, porém posteriormente se mantém.

Figura 19 – Boxplot da variância do repositório



Fonte: elaborada pelo autor

5.3.3 Análise do Conhecimento do Agente

Além da análise já realizada a respeito do Truck Factor e o repositório de software, outro aspecto importante que pode ser levado em consideração, é o conhecimento do agente. O conhecimento que pertence de forma individual ao agente, onde são armazenados todos as experiências obtidas durante o projeto. Com a conclusão da atividade, o agente ganha um adicional no conhecimento, que é relacionado ao aspecto da habilidade que foi necessário para realização da atividade. Essa nova perspectiva se faz necessária, visto que, o conhecimento individual obtido pelo agente é importante para o Truck Factor, já que nele buscamos encontrar a concentração de conhecimento.

Nesta análise, a fim de comparar as diferentes abordagens, iremos utilizar a variação do conhecimento geral dos agentes. Dessa forma podemos analisar de maneira geral os conhecimentos obtidos pelo agente nos diferentes aspectos do projeto. A variação do conhecimento do agente é a partir da variância da variável "knowledge_level" como apresentado na seção 4.4. Uma vez que buscamos fazer com que o conhecimento acerca do projeto seja distribuído entre todos os agentes envolvidos, a nossa métrica de variação do conhecimento precisa ser minimizada. Pois uma baixa variância, mostra que o conhecimento está distribuído entre os diferentes aspectos de conhecimento do projeto.

De maneira semelhante ao cálculo aplicado no repositório, para determinar a métrica do conhecimento do agente, foi necessário aplicar o cálculo de variância no atributos "knowledge level" de cada agente, então posteriormente foi realizado uma média dessa variância individual, logo obtemos nossa métrica. Esse processo foi realizado a cada do ciclo da sprint, da mesma maneira que já foi realizado nas outras métricas.

No algoritmo 5, podemos visualizar a implementação do pseudocódigo para a avaliação do conhecimento do agentes, notamos que a entrada é dada pelo conjunto de agentes que estão presente no ambiente, onde o algoritmo itera sobre cada agente realizando um cálculo simples de variância simples, no conjunto de dados "knowledge_level", então atribui esse valor para a variável "knowledge", onde posteriormente é realizada uma média simples. Ao final essa média é retornada como saída da função.

A partir dos resultados produzidos, obtemos a variância do conhecimento do agente nas diferentes abordagens, onde é apresentado na figura 20. Podemos observar como já apresentado nas outras análises, a abordagem Otimista permanece sendo a pior maneira de distribuir o conhecimento entre os membros. Em relação à abordagem GA e Random, notamos que a GA se

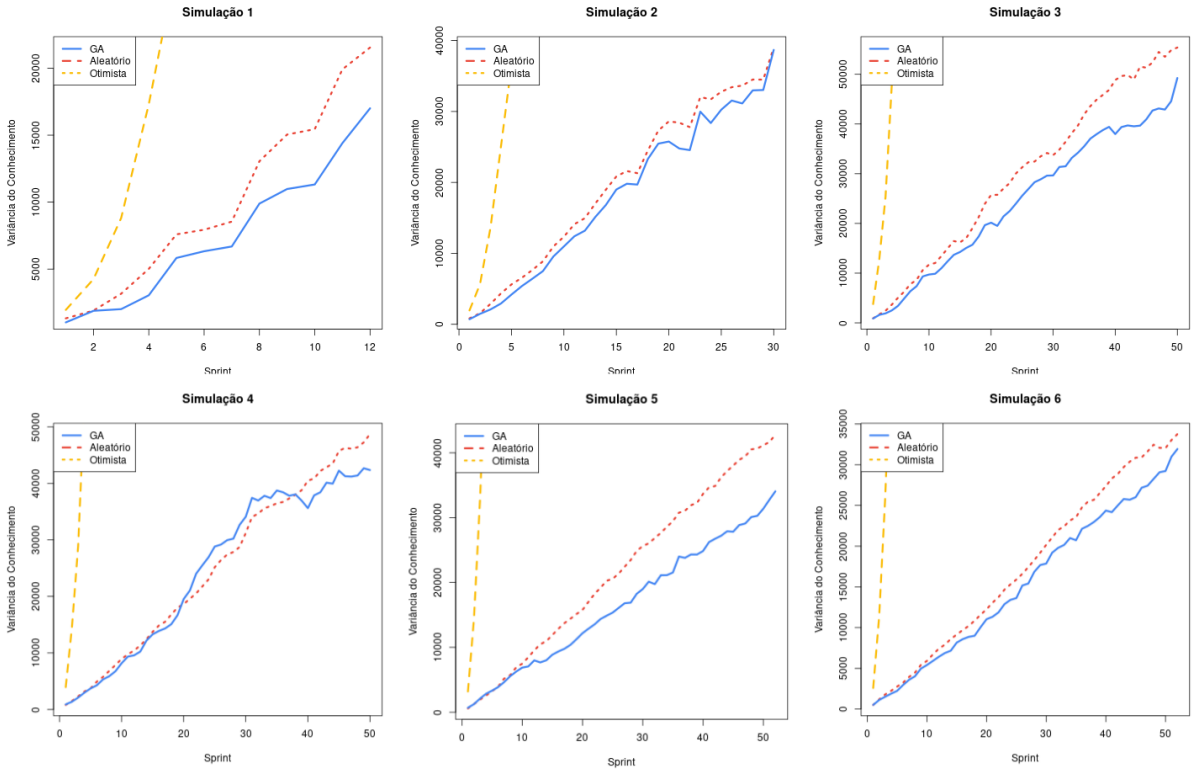
Algoritmo 5: Função de avaliação do conhecimento do agente

Input: Conjunto de Agentes no ambiente
Output: Média da variância do conhecimento dos agentes
$knowledge = []$
for $i \leftarrow 0$ *to* $len(agents)$ **do**
  $knowledge.add(variance(agent.knowledge\_level[i]))$;
end
return $mean(knowledge)$;

Fonte: elaborada pelo autor

mantém na maior parte dos cenários durante a simulação abaixo da Random, porém os resultados obtidos nessas duas abordagens são aproximados. De maneira positiva podemos destacar os cenários 1, 3 e 5, onde a abordagem GA tem um desempenho mais evidentes, e de maneira negativa, notamos o cenário 4, que por um momento da execução do projeto a abordagem GA obteve variância superior a Random, no entanto essa situação não se manteve ao final do projeto.

Figura 20 – Variância da variável de conhecimento do agente



Fonte: elaborada pelo autor

Na Tabela 9, é apresentada uma análise exploratória da variável de conhecimento do agente durante as sprints do projeto, por meio da tabela podemos mostrar um resumo de toda a execução do projeto, no que diz respeito à variação do conhecimento dos agentes envolvidos no

Tabela 9 – Análise exploratória da variância do conhecimento do agente

Simulação	Abordagem	Média	Mediana	Desvio Padrão
Simulação 1	GA	7534	6511	5211
	Aleatório	10040	8241	6880
	Otimista	56237	40290	50682
Simulação 2	GA	18236	19348	11300
	Aleatório	19956	21070	11739
	Otimista	367267	305164	306764
Simulação 3	GA	24643	26235	14055
	Aleatório	29805	31793	17075
	Otimista	2210628	1735083	1925519
Simulação 4	GA	24807	28990	14397
	Aleatório	25132	25785	15059
	Otimista	2906651	2232616	1925519
Simulação 5	GA	16638	16437	9601
	Aleatório	21909	21900	12949
	Otimista	3648167	2752861	3296107
Simulação 6	GA	15448	15288	9351
	Aleatório	17422	17012	10468
	Otimista	2856130	2148683	2579096

Fonte: elaborada pelo autor

projeto. Na tabela são apresentadas as médias, mediana e desvio padrão. Por meio da imagem apresentada na Figura 20, já notamos que as médias obtidas pela GA são inferiores a Random, porém através da tabela notamos que o desvio também é inferior, o que mostra que os resultados da GA são mais consistentes.

5.3.4 Teste de Normalidade

A Partir da análise de estatística descritiva, onde buscamos mostrar que a abordagem GA obteve melhor desempenho quando comparado com a Random e Otimista. A fim de compreender melhor o resultado obtido pelo Truck Factor e complementar a análise, apresentaremos um estudo realizado a partir de testes estatísticos. Nesta análise de testes estatísticos, como já mencionado, faremos uma análise somente a respeito da variável do Truck Factor, visto que a principal métrica investigada nesta pesquisa.

No primeiro momento precisamos compreender a respeito dos dados resultantes do Truck Factor. Para isso realizamos os testes de Kolmogorov Smirnov (KOLMOGOROV–SMIRNOV..., 2008) e Shapiro–Wilk (SHAPIRO; WILK, 1965). O teste é realizado em cima do conjunto de variáveis coletadas a partir de cada sprint durante o andamento do projeto, em cada cenário simulado. Os dois testes aplicados têm como objetivo mostrar se os resultados obtidos

Tabela 10 – Teste de Normalidade Kolmogorov Smirnov

Simulação	Abordagem	D	P-Value
Simulação 1	GA	0.53025	0.002346
	Aleatório		
	Otimista		
Simulação 2	GA	0.5391	5.344e-08
	Aleatório	0.40613	0.0001007
	Otimista	0.51678	2.197e-07
Simulação 3	GA	0.54007	4.302e-13
	Aleatório	0.52929	1.362e-12
	Otimista	0.48969	7.705e-11
Simulação 4	GA	0.52929	1.362e-12
	Aleatório	0.46007	1.284e-09
	Otimista	0.51385	6.819e-12
Simulação 5	GA	0.4949	1.733e-11
	Aleatório	0.36228	2.361e-06
	Otimista	0.46626	3.033e-10
Simulação 6	GA	0.39799	1.403e-07
	Aleatório	0.39198	2.297e-07
	Otimista	0.44738	1.825e-09

Fonte: elaborada pelo autor

representam ou não uma distribuição normal.

No teste de Kolmogorov a hipótese nula (H_0) é de que a amostra segue distribuição normal, a hipótese alternativa diz que é diferente de uma distribuição normal. Para mostrar que segue uma distribuição normal, temos $p > \alpha$, onde p é a probabilidade obtida da amostra e α mostra a significância do risco da amostra não seguir uma distribuição normal, para esta análise será aplicado o mesmo valor recomendado pela literatura, onde $\alpha = 0{,}05$ (5%).

Na Tabela 10 são apresentados os resultados obtidos pelo Kolmogorov em todos os cenários analisados, através da coluna (p-value), temos que os valores de p em todos os cenários apresentados são inferiores a $p < \alpha$, o que nos faz rejeitar a hipótese nula (H_0) de que a amostra segue uma distribuição normal, logo podemos inferir com 95% de confiabilidade que os dados não seguem uma distribuição normal.

Podemos destacar o resultado da simulação 1, em que as diferentes abordagens obtiveram os mesmo resultados do P-value. Isso se deu pelo fato do cenário da equipe ser bem pequeno, e os valores obtidos pelo Truck Factor não variaram durante o projeto, isso pode ser notado quando apresentado a Tabela 6, onde mostra um desvio padrão de 0 neste cenário.

O teste Shapiro tem como objetivo semelhante ao Kolmogorov, onde seu resultado mede o quão bem os dados se adequam a uma distribuição normal. O teste possui uma hipótese

Tabela 11 – Teste de Shapiro

Simulação	Abordagem	W	P-Value
Simulação 1	GA	-	-
	Aleatório	-	-
	Otimista	-	-
Simulação 2	GA	0.17962	7.766e-12
	Aleatório	0.61191	1.023e-07
	Otimista	0.4044	5.981e-10
Simulação 3	GA	0.19816	4.921e-15
	Aleatório	0.34386	1.3e-13
	Otimista	0.49037	6.231e-12
Simulação 4	GA	0.34386	1.3e-13
	Aleatório	0.5385	2.634e-11
	Otimista	0.38579	3.674e-13
Simulação 5	GA	0.43769	7.927e-13
	Aleatório	0.68868	3.256e-09
	Otimista	0.53884	1.538e-11
Simulação 6	GA	0.57799	5.473e-11
	Aleatório	0.49223	3.726e-12
	Otimista	0.56823	3.959e-11

Fonte: elaborada pelo autor

nula (H_0) de que a população pertence a uma distribuição normal e hipótese alternativa de que não seguem uma distribuição normal, onde $p \leq \alpha$, rejeitamos a H_0 e $p > \alpha$, não podemos rejeitar H_0. Para este teste também se utilizou do nível de significância $\alpha = 0,05$ (5%).

Na Tabela 11, são apresentados os resultados do teste de Shapiro, podemos observar que no cenário 1 não obteve saída, isso se deu pela situação que os valores do Truck Factor não variaram durante a execução do projeto, como já comentado anteriormente, essa situação ocorreu por conta do cenário relativamente pequeno. No entanto, observando os outros cenários seguintes, podemos observar que em todos os cenários analisados o valor de p foi inferior ao nosso α, dessa forma rejeitamos H_0, de que os dados seguem uma distribuição normal.

Ambos os testes estatísticos foram aplicados nesta pesquisa, a fim de complementar o que já se tinha como hipótese, de que os dados obtidos não seguem uma distribuição normal. Portanto, a partir dos resultados apresentados, podemos concluir com uma confiança de 95%, que existem evidências suficientes para rejeitar a hipótese de as variáveis do Truck Factor do projeto seguem uma distribuição normal.

5.3.5 Teste de Hipótese

Os testes de hipóteses fornecem evidências que nos permitem rejeitar ou não rejeitar uma hipótese estatística através da evidência fornecida pela amostra. A partir disso, objetivamos investigar as hipóteses de que a GA possui resultado superior à abordagem aleatória, no que diz respeito à distribuição de tarefas sob a métrica do Truck Factor. Comprovada tal hipótese, podemos comprovar estatisticamente o nosso resultado com um determinado grau de confiança.

O resultado do Teste de Wilcoxon (REY; NEUHÄUSER, 2011), nos permite determinar se as diferenças entre as medianas das 3 abordagens desenvolvidas, são estatisticamente significativas. Também nesse teste, utilizamos o grau de significância de $\alpha = 0,05$ (5%). O resultado obtido no Wilcoxon define-se onde, $p \leq \alpha$, então conclui-se que as medianas são significativamente diferentes, rejeitando a hipótese nula (H_0), caso $p > \alpha$, conclui-se que as medianas não são significativamente diferentes, então não podemos descartar a H_0.

O teste foi realizado comparando somente a abordagem GA e Aleatória, visto que foram pela estatística descritiva, as abordagens em alguns aspectos obtiveram resultados aproximados. O resultado foi gerado a partir de 3 diferentes hipóteses alternativas, onde comparada se a GA é Maior, Menor e Diferente da abordagem aleatória.

Na Tabela 12, são apresentados os resultados obtidos pelo teste de Wilcoxon, onde nota-se que em todos os cenários analisados, a hipótese alternativa da GA > Aleatória, resulta em um $p < \alpha$, o que nos faz rejeitar a hipótese nula. Na hipótese da GA $\neq$ Aleatória, também conseguimos rejeitar a hipótese nula em todos os cenários. No entanto, a hipótese da GA < Aleatória o $p > \alpha$, logo não devemos rejeitar a hipótese H_0.

Portanto, diante dos resultados apresentados, podemos concluir com um nível de confiança de 95%, que a mediana dos resultados obtidos pelo Truck Factor na abordagem GA são superiores aos obtidos na abordagem aleatória.

5.4 Considerações Finais

Neste capítulo foram apresentados os resultados obtidos nesta pesquisa. Inicialmente foram apresentados os resultados do ambiente simulado dentro do Netlogo, onde pode-se notar como são apresentados os agentes e as tarefas no ambiente, bem como os parâmetros utilizados para a configuração do ambiente, além disso foram mostrados os parâmetros de configuração do algoritmo genético, bem como sua função de avaliação do indivíduo. Posteriormente foram

Tabela 12 – Teste de WilCoxon abordagem GA e Aleatória

Simulação	Comparação	W	P-Value
Simulação 1	Maior	132	3.697e-05
	Menor		1
	Diferente		7.394e-05
Simulação 2	Maior	600	0.0007118
	Menor		0.9993
	Diferente		0.001424
Simulação 3	Maior	2455	2.2e-16
	Menor		1
	Diferente		2.2e-16
Simulação 4	Maior	2485	2.2e-16
	Menor		1
	Diferente		2.2e-16
Simulação 5	Maior	2704	2.2e-16
	Menor		1
	Diferente		2.2e-16
Simulação 6	Maior	2572.5	2.2e-16
	Menor		1
	Diferente		2.2e-16

Fonte: elaborada pelo autor

apresentados os estudos empíricos realizados no ambiente, onde se definiu alguns tipos de cenários que são utilizados para a realização de experimentos, e como também se deu o processo de coleta e compilação das variáveis coletadas. Por fim, foram apresentados os resultados da simulação nos diferentes ambientes, foram realizadas análises em 3 perspectivas, Truck Factor, repositório de software e conhecimento do agente. As análises através da estatística descritiva nos indicava que a abordagem GA obteve melhor desempenho, quando comparados com a Aleatória e Otimista, no entanto, a fim de complementar tal hipótese foram realizados testes estatísticos de normalidade e hipótese, onde concluiu-se o que já se tinha percebido na análise inicial, porém agora com um determinado grau de significância.

No apêndice A, podem ser consultados os dados brutos apresentados nesta seção. No apêndice C, pode ser consultado o script para gerar os resultados dos testes estatísticos.

No próximo capítulo serão apresentadas as considerações finais da pesquisa, considerando todos os resultados que foram coletados, bem como as contribuições realizadas, trabalhos futuros e as limitações desta pesquisa.

6 CONCLUSÕES

A execução de um projeto de software, é um processo complexo, que inclui diferentes etapas que precisam ser seguidas até sua conclusão. Do ponto de vista de gestão do software, é preciso construir um planejamento para mapear todos os passos necessários. Dentre as principais atividades de planejamento que precisam ser realizadas, está a alocação de tarefas, onde se mostra como processo complexo, devido às diferentes variáveis presentes no ambiente e suas restrições. A distribuição das atividades de forma otimizada, oferece garantias ao projeto como: redução de custos, entrega no prazo, delega atividade para as pessoas certas e etc.

Durante a execução do projeto, diferentes problemas podem surgir, se apresentando como uma ameaça ao plano do projeto. Dentre esses problemas, está a concentração de conhecimento. A partir disso, o Truck Factor tem o propósito de mensurar esses silos do conhecimento entre os membros da equipe. Por meio dessa métrica, pode-se compreender como o conhecimento a respeito do projeto está distribuído entre os membros da equipe, tem impacto direto no desempenho das entregas do projeto, como também pode representar uma ameaça a descontinuidade do mesmo. Buscar gerenciar tal métrica é de extrema importância, visto que a rotatividade de membros da equipe, que tem se tornado comum devido a globalização do mercado de TI.

Uma vez que o Truck Factor apresenta ameaça ao andamento do projeto, foi necessário estudar métodos para solucionar tal problemática, através de técnicas da SBSE viu-se que é possível aplicar uma vasta gama de algoritmos para encontrar uma solução ótima em um problema complexo no contexto da engenharia de software. De outro ponto de vista o ambiente de desenvolvimento se configura como um ambiente multiagente, visto que cada membro da equipe pode ser modelado como um agente com diferentes características, assim nesta pesquisa buscou-se realizar o estudo da distribuição de tarefas para os agentes visando minimizar os impactos causados pelo Truck Factor, a partir da abordagem de técnicas de SBSE.

Dentre os algoritmos que são utilizados em SBSE, nesta pesquisa utilizou-se a implementação de algoritmos genéticos que visem proporcionar uma solução ótima para a distribuição das tarefas. Visto que buscamos maximizar a distribuição de conhecimento do projeto entre os membros do time, a função de aptidão do algoritmo genético se baseou na contribuição do agente com o repositório, bem como no resultado fornecido pela métrica do Truck Factor.

Através do ambiente de simulação Netlogo, foi possível construir um ambiente

multiagente que visa incorporar variáveis do ambiente real de desenvolvimento, onde o usuário pode definir diferentes configurações de ambientes. No ambiente foram implementados diferentes agentes, com características e papéis pré-estabelecidos, no processo de alocação das tarefas, embora o ambiente multiagente proporciona uma função interna no agente para tomada de decisão, nesta pesquisa a distribuição das tarefas ficou com um agente centralizador, o Product Owner. O ambiente de simulação permitiu a validação da abordagem implementada nesta pesquisa, uma vez que o processo sendo executado no ambiente virtual, torna-se viável a reprodução e análise do ambiente em diferentes contextos, como também a coleta dos dados.

Os resultados apresentados mostraram que a abordagem desenvolvida com base no algoritmo genético obteve êxito na distribuição das tarefas, onde foi possível notar que embora as tarefas sejam geradas durante a construção do projeto, com tamanhos e características diferentes, a abordagem desenvolvida fez com que o Truck Factor se mantenha em um nível considerado satisfatório. Com o intuito de comparar a abordagem desenvolvida, foram implementadas outras duas estratégias de alocação de tarefas, a aleatória e a otimista. No entanto, através dos resultados apresentados, percebeu-se que as mesmas obtiveram um desempenho inferior.

A ferramenta desenvolvida tem como intuito auxiliar a gestão do projeto, para o monitoramento do risco do Truck Factor inerente ao projeto durante seu desenvolvimento, como também auxiliar o gestor para a alocação das tarefas, com foco na distribuição do conhecimento.

6.1 Contribuições

A partir do que se foi apresentado na discussão da proposta desenvolvida nesta pesquisa, e com o intuito de consolidar as ideias, lista-se abaixo as principais contribuições advindas da presente pesquisa:

- **Monitorar riscos ao projeto** Através da ferramenta proposta, é possível que o gestor do projeto ou o líder técnico a utilize como mecanismos de planejamento e distribuição das tarefas. Através do ambiente simulado, é possível construir diferentes cenários que possam representar o ambiente real da equipe. Por meio da ferramenta, será possível o gestor monitorar e mitigar os riscos que estão relacionados à gestão do conhecimento em relação ao projeto dentro da equipe.
- **Construção de ambiente multiagente baseado em Scrum:** Foram definidos através de diagramas a modelagem dos agentes do ambiente e por meio de diagramas de interação o comportamento e o papel de cada agente no ambiente, por meio disso foram implementados

um conjunto de componentes e fases que podem ser implantadas através de um ambiente multiagente. Por meio do ambiente virtual o usuário pode selecionar e configurar as variáveis no ambiente, através de parâmetros.

- **Implementação do ambiente e do algoritmo:** Através da ferramenta Netlogo, foi possível codificar o ambiente multiagente, com os diferentes tipos de agentes, papéis e atributos individuais. Por meio do Netlogo, também foi possível realizar a interação com a solução do algoritmo genético implementado em Python, onde o Netlogo ficou responsável pela construção do ambiente e o controle dos agentes e o algoritmo genético tem a função de buscar a solução para o problema de alocação. Para o algoritmo genético foi elaborado uma função de aptidão do indivíduo que busque gerar uma população que maximize a métrica do Truck Factor.
- **Truck Factor em diferentes ambientes e abordagens:** Uma vez que o ambiente para estudar os impactos do Truck Factor foi implementado em um ambiente simulado, foi possível realizar um estudo sobre diferentes abordagens e um ambiente com características distintas. Onde foi possível criar desde um ambiente com poucas pessoas na equipe, o que representa um projeto de pequeno porte, até projetos que possuem uma grande quantidade de pessoas e tarefas a serem alocadas.
- **Métricas para análise de risco do projeto:** Na pesquisa foi utilizada a métrica do Truck Factor com o objetivo de validação da abordagem implementada, no entanto, foram elaboradas outros mecanismos para avaliação do repositório de software, como também a avaliação sobre o conhecimento individual do agente. Todas essas métricas desenvolvidas, foram utilizadas também com objetivo de validação da abordagem, porém analisam a concentração de conhecimento de uma perspectiva diferente.

6.2 Limitações da Pesquisa

Acredita-se que os objetivo da pesquisa foram atingidos, com resultado satisfatório, no entanto, algumas limitações podem ser observadas:

- **Ausência de ambientes reais** Uma das principais razões para o experimento ser realizado em um ambiente de simulação multiagente, foi a falta de um ambiente real de desenvolvimento de software que pudesse ser gerenciado pela abordagem proposta e analisando seus resultados. Além disso seria mais complexo, visto que a coleta das variáveis do ambiente seriam limitadas a poucos cenários.

- **Limitações das variáveis** O ambiente real de um projeto de desenvolvimento de software é complexo e possui diversas variáveis que podem gerar impacto no projeto, no entanto na pesquisa, devido ao ambiente simulado, foram implementadas as variáveis mais objetivas que tem relação ao projeto. No entanto, analisar outros aspectos do projeto, torna-se fundamental para a ferramenta.
- **Escassez de estratégias para alocação** Embora no ambiente virtual seja possível criar diferentes propostas para distribuição das tarefas, foram implementadas apenas outras duas abordagens para serem comparados os resultados. A ausência dessas outras implementações, deixa uma questão em saber se a GA seria a mais eficiente quando comparado com outros algoritmos de otimização, para a solução da alocação de tarefas.

6.3 Trabalhos Futuros

A partir da implementação desta proposta, ainda se faz necessário o aperfeiçoamento e continuidade do trabalho, incluindo as próprias limitações já observadas:

- **Implementar ferramenta** A construção de uma ferramenta com interface gráfica adequada ao usuário que se baseia na proposta apresentada é fundamental para torná-la uma ferramenta de acesso aos gestores de projetos de software. Uma vez que na sua implementação surgirão novos desafios a serem resolvidos, como a análise de novas variáveis que o ambiente possui, bem como também a integração com ferramentas de gerenciamento de projeto, como: Trello [1], Jira [2] e etc. No entanto, a partir da construção da ferramentas, gestores de projetos têm ao seu auxílio uma ferramenta que sugere a alocação de tarefas, buscando maximizar a métrica da distribuição do conhecimento entre os membros da equipe.
- **Implantar e realizar experimentos** Através do modelo de simulação proposta nesta pesquisa, é possível realizar alguns ajustes no modelo e implantá-lo em um ambiente real de desenvolvimento de software, onde os gestores e líderes possam realizar estudos e análises sobre sua equipe, dessa maneira avaliando e podendo sugerir ajustes na ferramenta proposta.
- **Propor novos algoritmos** Outro objetivo que deve ser atingido com a pesquisa, será a implementação de novos algoritmos para serem utilizados na alocação das tarefas. Essa

[1] Acesse em: https://www.trello.com
[2] Acesse em: https://www.atlassian.com/software/jira

nova perspectiva se faz necessária, visto que com a implantação da ferramenta no ambiente real, torna-se interessante para a gestão utilizar abordagens distintas de algoritmos de otimização para distribuir as tarefas.

– **Propor algoritmo para o Truck Factor** Além do algoritmo de otimização, o cálculo do Truck Factor é outra oportunidade de melhoria, visto que na pesquisa o Truck Factor se baseou somente no ponto de vista do repositório de software, e no ambiente real, outras variáveis subjetivas podem ser levadas em consideração. Além de que nesta área de pesquisa, outras propostas para determinar a métrica do Truck Factor ainda estão em aberto.

REFERÊNCIAS

ABBAS, H. A.; SHAHEEN, S. I.; AMIN, M. H. Organization of multi-agent systems: an overview. **arXiv preprint arXiv:1506.09032**, 2015.

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.

ARAÚJO, A. A. d. P. Uma arquitetura utilizando algoritmo genético interativo e aprendizado de máquina aplicado ao problema do próximo release. 2015.

AVELINO, G.; PASSOS, L.; HORA, A.; VALENTE, M. T. A novel approach for estimating truck factors. In: IEEE. **2016 IEEE 24th International Conference on Program Comprehension (ICPC)**. [*S. l.*], 2016. p. 1–10.

BABIŠ, M.; MAGULA, P. Netlogo—an alternative way of simulating mobile ad hoc networks. In: IEEE. **2012 5th Joint IFIP Wireless and Mobile Networking Conference (WMNC)**. [*S. l.*], 2012. p. 122–125.

BAIA, D. de M. An integrated multi-agent-based simulation approach to support software project management. In: IEEE. **2015 IEEE/ACM 37th IEEE International Conference on Software Engineering**. [*S. l.*], 2015. v. 2, p. 911–914.

BALAJI, P.; SRINIVASAN, D. An introduction to multi-agent systems. In: **Innovations in multi-agent systems and applications-1**. [*S. l.*]: Springer, 2010. p. 1–27.

BANSALL, S. **Agents in Artificial Intelligence**. 2020. https://www.geeksforgeeks.org/agents-artificial-intelligence/. Accessed: 2020-09-28.

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

CHAE, S. W.; SEO, Y. W.; LEE, K. C. Task difficulty and team diversity on team creativity: Multi-agent simulation approach. **Computers in Human Behavior**, Elsevier, v. 42, p. 83–92, 2015.

CHUMACHENKO, D.; MENIAILOV, I.; BAZILEVYCH, K.; CHUKHRAY, A. Intelligent multiagent approach to diphtheria infection epidemic process simulation. In: IEEE. **2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)**. [*S. l.*], 2019. p. 833–836.

DARWIN, C. On the origin of species by means of natural selection. 1859. **London: Murray Google Scholar**, 1968.

DEAP. **Distributed Evolutionary Algorithims in Python**. 2022. Disponível em: https://deap.readthedocs.io/en/master/.

DEMIROVIĆ, E.; SCHWIND, N.; OKIMOTO, T.; INOUE, K. Recoverable team formation: Building teams resilient to change. In: **Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems**. [*S. l.*: *s. n.*], 2018. p. 1362–1370.

EBADI, T.; PURVIS, M.; PURVIS, M. Partner selection mechanisms for agent cooperation. In: IEEE. **2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology**. [*S. l.*], 2008. v. 3, p. 554–557.

FARHANGIAN, M.; PURVIS, M.; PURVIS, M.; SAVARIMUTHU, T. B. R. Agent-based modeling of resource allocation in software projects based on personality and skill. In: SPRINGER. **International Workshop on Multiagent Foundations of Social Computing**. [*S. l.*], 2015. p. 130–146.

FARHANGIAN, M.; PURVIS, M.; PURVIS, M.; SAVARIMUTHU, B. T. R. Personalities and software development team performance, a psycholinguistic study. In: **24th European Conference on Information Systems**. [*S. l.*: *s. n.*], 2016.

FAROOQ, H.; JANJUA, U. I.; MADNI, T. M.; WAHEED, A.; ZAREEI, M.; ALANAZI, F. Identification and analysis of factors influencing turnover intention of pakistan it professionals: An empirical study. **IEEE Access**, IEEE, 2022.

FERREIRA, M.; AVELINO, G.; VALENTE, M. T.; FERREIRA, K. A. A comparative study of algorithms for estimating truck factor. In: IEEE. **2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)**. [*S. l.*], 2016. p. 91–100.

FERREIRA, M.; VALENTE, M. T.; FERREIRA, K. A comparison of three algorithms for computing truck factors. In: IEEE. **2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)**. [*S. l.*], 2017. p. 207–217.

HANNEBAUER, C.; GRUHN, V. Algorithmic complexity of the truck factor calculation. In: SPRINGER. **International Conference on Product-Focused Software Process Improvement**. [*S. l.*], 2014. p. 119–133.

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

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. **Department of Computer Science, King's College London, Tech. Rep. TR-09-03**, p. 23, 2009.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search-based software engineering: Trends, techniques and applications. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 45, n. 1, p. 1–61, 2012.

HILTON, M.; BEGEL, A. A study of the organizational dynamics of software teams. In: **Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice**. [*S. l.*: *s. n.*], 2018. p. 191–200.

HONGQIAO, Y.; XIHUA, L.; FEI, W.; WEIZI, L. Multi-agent based modeling and simulation of complex system in hospital. In: IEEE. **2009 16th International Conference on Industrial Engineering and Engineering Management**. [*S. l.*], 2009. p. 1759–1763.

JABRAYILZADE, E.; EVTIKHIEV, M.; TÜZÜN, E.; KOVALENKO, V. Bus factor in practice. **arXiv preprint arXiv:2202.01523**, 2022.

KOLMOGOROV–SMIRNOV Test. In: THE Concise Encyclopedia of Statistics. New York, NY: Springer New York, 2008. p. 283–287. ISBN 978-0-387-32833-1. Disponível em: https://doi.org/10.1007/978-0-387-32833-1_214.

KROTHAPALLI, N.; DESHMUKH, A. V. Distributed task allocation in multi-agent systems. In: **Proceedings of the Institute of Industrial Engineers Annual Conference**. [*S. l.*: *s. n.*], 2002.

LAMBORA, A.; GUPTA, K.; CHOPRA, K. Genetic algorithm-a literature review. In: IEEE. **2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)**. [*S. l.*], 2019. p. 380–384.

LI, K.; WU, S.; WEN, Y.; WANG, Y. Task allocation of multiagent groups in social network ed systems. **IEEE Internet of Things Journal**, IEEE, 2021.

LI, M.; WANG, Z.; LI, K.; LIAO, X.; HONE, K.; LIU, X. Task allocation on layered multiagent systems: When evolutionary many-objective optimization meets deep q-learning. **IEEE Transactions on Evolutionary Computation**, IEEE, v. 25, n. 5, p. 842–855, 2021.

MAHESH, P.; ANKITHA, N.; TARAKESH, M.; DHARAVATH, K.; AMARNATH, G. A simulation study of covid-19 out break as an epidemic disease. In: IEEE. **2022 International Conference on Computer Communication and Informatics (ICCCI)**. [*S. l.*], 2022. p. 1–5.

Miller, W.; Spooner, D. L. Automatic generation of floating-point test data. **IEEE Transactions on Software Engineering**, SE-2, n. 3, p. 223–226, 1976.

MITCHELL, M. **An introduction to genetic algorithms**. [*S. l.*]: MIT press, 1998.

MOHAMAD, M. R.; NASARUDDIN, F. H.; HAMID, S.; BUKHARI, S.; IJAB, M. T. Predicting employees' turnover in it industry using classification method with feature selection. In: IEEE. **2021 International Conference on Computer Science and Engineering (IC2SE)**. [*S. l.*], 2021. v. 1, p. 1–7.

NETLOGO. **NetLogo is a multi-agent programmable modeling environment.** 2022. Disponível em: https://ccl.northwestern.edu/netlogo/.

PAIXAO, M.; SOUZA, J. A robust optimization approach to the next release problem in the presence of uncertainties. **Journal of Systems and Software**, Elsevier, v. 103, p. 281–295, 2015.

PEE, L. G.; KANKANHALLI, A.; TAN, G. W.; THAM, G. Mitigating the impact of member turnover in information systems development projects. **IEEE Transactions on Engineering Management**, IEEE, v. 61, n. 4, p. 702–716, 2014.

PETROVIĆ, N. Simulation environment for optimal resource planning during covid-19 crisis. In: IEEE. **2020 55th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)**. [*S. l.*], 2020. p. 23–26.

PYTHON. **Python is a programming language that lets you work quickly and integrate systems more effectively**. 2022. Disponível em: https://www.python.org/.

RÄIHÄ, O. A survey on search-based software design. **Computer Science Review**, Elsevier, v. 4, n. 4, p. 203–249, 2010.

RAILSBACK, S. F.; LYTINEN, S. L.; JACKSON, S. K. Agent-based simulation platforms: Review and development recommendations. **Simulation**, Sage Publications Sage CA: Thousand Oaks, CA, v. 82, n. 9, p. 609–623, 2006.

REPAST. **The Repast Suite is a family of advanced, free, and open source agent-based modeling and simulation platforms.** 2020.

REY, D.; NEUHÄUSER, M. Wilcoxon-signed-rank test. In: ____. **International Encyclopedia of Statistical Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 1658–1659. ISBN 978-3-642-04898-2. Disponível em: https://doi.org/10.1007/978-3-642-04898-2_616.

ROJAS, J. A.; GIACHETTI, R. E. An agent-based simulation model to analyze team performance on jobs with a stochastic structure. In: IEEE. **2009 First International Conference on Advances in System Simulation**. [*S. l.*], 2009. p. 148–154.

RUBIN, K. S. **Essential Scrum: A practical guide to the most popular Agile process**. [*S. l.*]: Addison-Wesley, 2012.

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach. 2002.

SANDIM, H.; BRANDAO, M. A.; MORO, M. M. Stf: uma abordagem social para estimar truck factor no github.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, Oxford University Press (OUP), v. 52, n. 3-4, p. 591–611, dec 1965. Disponível em: https://doi.org/10.1093/biomet/52.3-4.591.

SIEBERS, P.-O.; AICKELIN, U. Introduction to multi-agent simulation. In: **Encyclopedia of decision making and decision support technologies**. [*S. l.*]: IGI Global, 2008. p. 554–564.

SINGHAL, V.; DAHIYA, D. Distributed task allocation in dynamic multi-agent system. In: IEEE. **International Conference on Computing, Communication & Automation**. [*S. l.*], 2015. p. 643–648.

SOMMERVILLE, I. **Software Engineering**. 9th. ed. USA: Addison-Wesley Publishing Company, 2010. ISBN 0137035152.

SOMMERVILLE, I. **Engenharia de Software**. Third. São Paulo: Pearson Prentice Hal, 2011.

SPADE. **A multi-agent systems platform written in Python and based on instant messaging (XMPP).** 2020. Disponível em: https://pypi.org/project/spade/.

SUTHERLAND, J. **Scrum: The Art of Doing Twice the Work in Half the Time**. Random House Business Books, 2015. ISBN 9781847941107. Disponível em: https://books.google.com.br/books?id=L13frQEACAAJ.

TONELLA, P.; SUSI, A.; PALMA, F. Using interactive ga for requirements prioritization. In: IEEE. **2nd International Symposium on Search Based Software Engineering**. [*S. l.*], 2010. p. 57–66.

WARCHOLINSKI, M. **Lean, Agile and Scrum: A Simple Guide**. 2020. https://brainhub.eu/blog/differences-lean-agile-scrum/. Accessed: 2020-09-26.

WOOLDRIDGE, M. **An introduction to multiagent systems**. [*S. l.*]: John Wiley & Sons, 2009.

YASIR, M.; PURVIS, M.; PURVIS, M.; SAVARIMUTHU, B. T. R. Complementary-based coalition formation for energy microgrids. **Computational Intelligence**, Wiley Online Library, v. 34, n. 2, p. 679–712, 2018.

ZAZWORKA, N.; STAPEL, K.; KNAUSS, E.; SHULL, F.; BASILI, V. R.; SCHNEIDER, K. Are developers complying with the process: an xp study. In: **Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**. [*S. l.*: *s. n.*], 2010. p. 1–10.

ZHANG, Y.; ZHOU, Y.; CAO, J. Research and simulation of ant colony foraging behavior based on netlogo. In: IEEE. **2022 International Conference on Computation, Big-Data and Engineering (ICCBE)**. [*S. l.*], 2022. p. 177–180.

ZHOU, Y.; CAO, J.; ZHANG, Y. Research and simulation of computer virus propagation model based on netlogo. In: IEEE. **2022 International Conference on Computation, Big-Data and Engineering (ICCBE)**. [*S. l.*], 2022. p. 13–17.

APÊNDICE A – DADOS COMPILADOS

Tabela 13 – Cenário 1

N	VG	VR	VO	KG	KA	KO	TG	TA	TO
1	117	167	149	1022	1322	1955	2	1	1
2	173	300	336	1885	1903	4249	2	1	1
3	104	447	649	2011	3160	8809	2	1	1
4	127	494	901	3045	5016	17312	2	1	1
5	99	546	1165	5832	7594	27664	2	1	1
6	120	634	1316	6333	7938	36010	2	1	1
7	135	654	1462	6688	8544	44569	2	1	1
8	170	747	1651	9891	13053	64754	2	1	1
9	168	991	1483	10979	15045	87504	2	1	1
10	155	1036	1690	11318	15448	100646	2	1	1
11	204	1176	1805	14404	19915	130830	2	1	1
12	228	1189	1964	17005	21543	150543	2	1	1

Tabela 14 – Caso 2

N	VG	VR	VO	KG	kA	KO	TG	TA	TO
1	60.16	71	70	710.63	836	1958	3	2	2
2	73.69	119	183	1467.46	1559	5636	3	2	2
3	95.08	171	371	2090.87	2871	13596	3	2	2
4	101.79	224	620	2929.41	4350	25078	3	2	2
5	105.11	257	774	4188.33	5617	36495	3	3	2
6	110.08	300	887	5392.76	6564	49016	3	3	2
7	125.22	354	1050	6428.61	7661	67281	3	2	2
8	131.61	386	1271	7498.52	8873	86630	3	3	2
9	138.27	404	1501	9560.61	10973	110915	3	3	2
10	159.7	440	1641	10975.53	12351	136647	3	3	2
11	159.04	468	1722	12405.29	14111	164982	3	3	2
12	152.5	474	1755	13197.09	14958	189431	3	3	2
13	151.88	485	1780	15137.24	17037	221550	3	2	2
14	144.3	472	1779	16817.39	18973	249844	3	2	2
15	148.91	489	1918	18998.9	20841	291591	3	3	2
16	142.53	490	2015	19810.06	21603	318736	3	3	2
17	151.35	512	2145	19696.94	21299	354352	3	2	2
18	153.31	515	2139	23291.88	24499	402746	3	2	2
19	166.5	527	2231	25464.35	27310	447602	3	3	2
20	156.7	534	2287	25759.46	28598	487291	3	3	2
21	148.94	522	2180	24779.28	28419	511808	3	3	2
22	166.58	543	2219	24551.72	27790	552661	3	3	2
23	170.35	523	2345	29970.91	32033	616892	3	3	2
24	173.07	519	2401	28368.31	31707	654959	3	3	1
25	167.78	507	2425	30250.83	32780	699703	3	3	2
26	167.5	517	2487	31538.58	33421	749211	3	3	1
27	166.99	517	2489	31131.45	33647	788550	3	3	2
28	181.49	555	2601	32970.93	34503	863800	3	3	1
29	183.47	560	2541	33026.24	34488	914582	3	2	2
30	201.63	589	2553	38674.57	39013	1004472	3	2	1

Tabela 15 – Caso 3

N	VG	VR	VO	KG	kA	KO	TG	TA	TO
1	51.16	58	68	874.02	904	3783	4	3	2
2	83.13	103	218	1607.08	1717	12842	5	3	2
3	106.25	138	362	1904.45	2428	25367	4	4	2
4	128.3	186	569	2487.79	3647	47765	5	4	2
5	158.78	226	807	3407.69	5039	76061	5	4	2
6	171.12	263	1034	4886.2	6262	112863	5	4	2
7	177.57	297	1260	6362.1	7785	151047	5	3	1
8	175.01	314	1408	7371.71	8646	188086	5	3	2
9	175.41	345	1599	9338.33	10614	241084	5	4	2
10	186.3	361	1719	9724.02	11710	290732	5	3	2
11	176.46	352	1846	9871.64	12037	325592	5	4	2
12	180.56	372	2026	10992.99	13446	393211	5	4	1
13	181.51	387	2196	12376.36	14969	462499	5	4	1
14	192.69	396	2365	13679.5	16446	538773	5	4	1
15	199.97	407	2589	14252.94	16194	606694	5	4	1
16	209.3	421	2691	15070.22	17048	686368	5	4	1
17	222.16	444	2952	15715.52	19045	786656	5	4	1
18	223.69	455	3251	17362.6	21268	880401	5	4	1
19	231.82	471	3442	19654.71	23951	998719	5	4	1
20	229.93	475	3487	20146.98	25725	1088916	5	4	1
21	224.66	477	3475	19503.77	25758	1174149	5	4	1
22	235.16	496	3682	21408.17	27014	1303738	5	4	1
23	228.91	492	3821	22492.53	28115	1417121	5	4	1
24	232.6	510	3983	23982.33	30045	1545525	5	4	1
25	231.38	514	4148	25556.93	31277	1668968	5	4	1
26	234.31	523	4197	26912.6	32309	1801198	5	4	1
27	247.47	525	4348	28287.32	32439	1939138	5	4	1
28	257.12	530	4523	28882.25	33506	2079568	5	4	1
29	249.38	530	4538	29606.76	34161	2210732	5	4	1
30	252	528	4727	29650.72	33740	2327926	5	4	1
31	250.8	527	4786	31359.08	34841	2473571	5	4	1
32	264.47	541	5048	31491.69	36416	2664308	5	4	1
33	270.94	551	5253	33178.07	38149	2832667	5	4	1
34	281.94	553	5430	34229.8	39714	3014774	5	4	1
35	274.38	548	5424	35523.59	41952	3197000	5	4	1
36	256.46	537	5463	37067.2	43605	3348693	5	4	1
37	257.12	541	5588	37953.4	44892	3553196	5	4	1
38	244.98	528	5496	38771.55	45848	3703805	5	4	1
39	257.02	537	5774	39423.64	46844	3907490	5	4	1
40	248.71	540	5724	37973.32	48845	4096507	5	4	1
41	243.72	524	5640	39381.75	49647	4254662	5	4	1
42	245.76	526	5859	39709.87	49836	4447087	5	4	1
43	243.25	522	5866	39538	48954	4619976	5	4	1
44	243.43	538	6011	39674.17	51552	4876660	5	4	1
45	243.03	529	5960	40937.91	51322	5068044	5	4	1
46	251.64	538	6229	42706.55	52378	5324629	5	4	1
47	257.46	542	6176	43122.8	54480	5578121	5	4	1
48	247.79	543	6228	42903.92	53527	5768300	5	4	1
49	244.53	555	6180	44577.76	54841	6056053	5	4	1
50	251.98	587	6313	49248.37	55382	6360313	5	4	1

Tabela 16 – Caso 4

N	VG	VR	VO	KG	kA	KO	TG	TA	TO
1	46.98	52	61	896.84	802	3954	6	4	2
2	73.64	92	167	1365.19	1495	14536	7	4	3
3	95.21	126	287	2100.43	2307	28908	6	5	3
4	130.74	163	437	2961.41	3164	54506	6	5	2
5	147.76	191	595	3716.35	3738	80150	7	5	2
6	176.75	231	769	4269.94	4914	121043	7	5	2
7	182.42	248	888	5307.18	5752	158219	7	5	1
8	198.46	282	1076	5874.48	6810	211742	6	5	1
9	214.37	303	1230	6752.6	7857	271147	7	5	1
10	232.21	334	1411	8172.54	8943	346138	7	5	1
11	238.84	353	1584	9346.66	9743	415940	7	5	1
12	239.47	362	1740	9570.44	10419	487767	7	5	1
13	239.19	364	1870	10230.17	11311	559302	7	5	1
14	248.58	376	1988	12178.15	12424	649019	7	5	1
15	260.98	393	2205	13292.4	13735	753136	7	5	1
16	271.64	405	2336	13881.24	14840	861633	6	5	1
17	263.97	416	2448	14322.83	15505	961754	7	5	1
18	270.03	428	2554	15078.98	16793	1081653	7	5	1
19	276.23	433	2700	16664.75	18008	1213800	7	5	1
20	273.66	435	2855	19538.44	18641	1335508	7	5	1
21	266.03	439	2925	21056.9	19584	1467888	7	6	1
22	277.16	450	3011	24008.07	20522	1623732	7	6	1
23	277.83	461	3063	25495.27	21761	1781757	7	6	1
24	285	473	3175	26926.03	23004	1961673	7	6	1
25	294.8	491	3410	28815.51	25159	2152287	7	5	1
26	293.44	494	3431	29164.81	26410	2312945	7	5	1
27	302.96	497	3424	29942.31	27394	2496097	7	5	1
28	300.39	498	3567	30210.41	27844	2659953	7	5	1
29	297.4	498	3564	32647.05	28725	2833533	7	5	1
30	306.6	516	3673	34099.04	31224	3065330	7	5	1
31	305.53	519	3768	37425.86	34044	3306587	7	5	1
32	303.54	520	4035	36952.92	34641	3511310	7	5	1
33	300.33	510	4046	37792.25	35603	3710368	7	5	1
34	280.07	496	3917	37386.84	35964	3902139	7	5	1
35	291.08	512	4013	38726.53	36467	4167177	7	5	1
36	291.28	513	4087	38414.6	36698	4379936	7	5	1
37	299.06	517	4230	37799.72	37325	4643546	7	5	1
38	304.58	519	4345	38040.6	37987	4900861	7	5	1
39	297.73	508	4324	36933.14	38790	5113728	7	5	1
40	301	518	4453	35608.55	40463	5413021	7	5	1
41	298.18	521	4480	37862.79	40851	5700116	7	5	1
42	296.59	523	4562	38406.35	42182	5986449	7	5	1
43	295.64	520	4552	40111.21	42840	6267647	7	5	1
44	295.03	518	4665	39942.01	43488	6580053	7	5	1
45	306.02	519	4721	42239.66	45727	6917012	7	5	1
46	304.11	519	4718	41254.47	46272	7207663	7	5	1
47	288.8	506	4634	41197.84	46116	7450496	7	6	1
48	285.29	502	4587	41381.11	46372	7730344	7	5	1
49	283.33	499	4567	42662.63	47244	8053199	7	5	1
50	294.18	516	4750	42346.11	48705	8425852	7	6	1

Tabela 17 – Caso 5

N	VG	VR	VO	KG	kA	KO	TG	TA	TO
1	36.32	39	48	686.16	586	3193	8	5	2
2	71.48	84	176	1242.92	1261	14431	8	6	2
3	103.46	121	351	2094.24	1941	32714	9	6	2
4	129.13	151	492	2837.35	2559	56539	9	7	2
5	146.59	174	641	3321.43	3241	86365	9	6	2
6	157.5	196	831	3867.73	3925	122656	9	6	2
7	173.33	231	1059	4639.5	5184	173792	8	7	1
8	187.71	249	1241	5546.17	5705	221710	9	7	2
9	214.17	278	1461	6260.66	6913	295688	9	7	2
10	217.82	298	1641	6872.52	7477	360615	9	6	2
11	229.05	315	1802	7051.16	8459	436902	9	6	2
12	236.84	332	2009	7987.99	9534	523257	9	6	1
13	248.66	345	2164	7677.28	10473	612202	9	6	2
14	262.57	358	2321	8006.08	10944	711226	9	7	2
15	274.57	371	2553	8863.97	11899	822320	9	7	2
16	291.02	387	2846	9362.16	12860	946510	9	7	1
17	291.42	397	3049	9778	13725	1064377	9	6	1
18	299.58	411	3278	10379.05	14406	1211310	9	7	1
19	299.46	422	3470	11250.23	15125	1355078	9	6	1
20	302.09	442	3706	12176.13	15765	1516705	9	6	1
21	304.27	460	3867	12876.36	17026	1695981	9	6	1
22	306.25	464	4085	13534.62	18231	1852628	9	6	1
23	315.69	470	4301	14366.51	19194	2038505	9	6	1
24	321.87	474	4447	14868.46	20261	2233917	9	6	1
25	327.65	481	4510	15349.61	20565	2443480	9	6	1
26	329.24	484	4615	16076.08	21366	2660806	9	6	1
27	320.67	485	4740	16797.05	22433	2844916	9	6	1
28	325.97	492	4866	16885.32	23465	3072547	9	6	1
29	325.13	493	5017	18254.8	24863	3306895	9	7	1
30	313.45	491	5076	18982.32	25616	3523250	8	7	1
31	314.81	492	5159	20129.66	26031	3773915	9	7	1
32	315.1	491	5350	19741.47	26852	4012818	9	6	1
33	322.34	501	5467	21124.19	27646	4291059	9	6	1
34	328.57	505	5606	21128.28	28582	4582980	9	7	1
35	321.88	505	5678	21539.79	29456	4842857	9	7	1
36	329.11	512	5831	24000.9	30754	5151609	9	7	1
37	323.13	516	5838	23794.3	31060	5441022	9	7	1
38	323.05	505	5842	24316.29	31909	5684873	9	7	1
39	327.58	506	5922	24313.32	32279	5976124	9	7	1
40	342.4	511	6127	24851.08	33659	6326650	9	6	1
41	343.36	521	6338	26224.75	34663	6693795	8	7	1
42	334.37	516	6323	26738.66	34869	7003575	9	7	1
43	340.4	527	6341	27200.19	36212	7376898	9	7	1
44	332.01	524	6408	27894.7	37178	7723338	9	6	1
45	328.15	508	6224	27818.22	38018	8003055	9	7	1
46	331.08	507	6299	28849.82	38904	8389561	9	6	1
47	331.99	508	6403	29086.42	39452	8777817	9	6	1
48	328.79	503	6372	30087.3	40523	9114630	10	6	1
49	315.43	498	6279	30273.85	40655	9435228	9	6	1
50	320.76	509	6320	31376.94	41146	9862196	9	6	1
51	323.59	515	6295	32745.51	41652	10281490	9	6	1
52	337.62	521	6470	34057.56	42779	10718694	9	6	1

Tabela 18 – Caso 6

N	VG	VR	VO	KG	kA	KO	TG	TA	TO
1	15.04	16	18	510.11	506	2589	6	4	2
2	31.38	33	63	1110.97	1190	11353	9	7	2
3	42.94	49	118	1509.98	1789	25762	8	8	2
4	51.44	59	160	1883.59	2232	44371	10	8	2
5	57.44	68	212	2228.01	2753	66916	9	8	2
6	64.2	77	269	2943.98	3236	96017	11	9	2
7	73.21	88	338	3586.57	3933	136111	10	8	2
8	77.11	95	389	4023.25	4471	173606	10	8	2
9	85.17	106	444	4963.53	5420	232474	11	8	2
10	87.31	111	490	5402.93	5904	283090	10	9	2
11	89.95	116	542	5905.24	6696	339932	10	8	2
12	95	121	593	6414.28	7410	408092	11	8	2
13	97.27	124	636	6868.71	7893	477162	11	9	2
14	100.63	130	672	7162.9	8596	555112	11	9	2
15	103.48	134	726	8156.23	9143	641751	10	9	2
16	108.26	141	772	8577.32	9693	743920	10	9	2
17	109.63	144	820	8859.74	10200	837790	10	9	2
18	114.69	148	890	8996.68	10849	951143	11	9	2
19	117.17	151	925	10050.04	11577	1064359	11	9	2
20	120.55	157	992	11027.21	12247	1189647	11	8	2
21	125.1	162	1055	11324.77	13058	1325245	11	8	2
22	124.21	163	1074	11866.32	13733	1453839	11	8	2
23	126.25	165	1112	12874.33	14653	1604190	11	8	2
24	128.88	169	1168	13402.43	15298	1751770	11	9	2
25	130.41	172	1219	13626.07	15899	1908187	11	9	2
26	129.54	173	1252	15167.27	16599	2075735	11	9	2
27	127.93	172	1267	15408.32	17425	2221631	11	9	2
28	128.86	173	1295	16841.97	18289	2401452	11	9	2
29	130.15	174	1326	17679.1	19196	2584040	11	9	2
30	127.73	172	1335	17857.91	20155	2752759	11	9	2
31	129.27	175	1359	19195.41	21057	2948046	11	9	2
32	128.51	176	1355	19790.43	21987	3146367	11	9	2
33	129.9	178	1373	20147.29	22429	3366123	11	9	2
34	132.66	181	1406	20991.6	23155	3595953	10	9	2
35	131.45	180	1433	20724.23	23550	3797886	11	9	2
36	135.19	184	1464	22132.82	24760	4038243	11	9	2
37	134.1	184	1473	22486.29	25460	4259800	11	9	2
38	129.97	180	1449	22966	25692	4456182	11	9	1
39	128.51	181	1461	23574.3	26528	4684866	11	9	1
40	131.97	184	1503	24365.23	27359	4961831	11	9	1
41	134.52	189	1538	24173.13	28309	5250900	11	9	1
42	135.19	187	1536	24995.03	28857	5492192	12	9	1
43	137.25	187	1568	25786.51	29710	5776736	11	9	1
44	137.01	186	1565	25695.58	30364	6054526	11	9	1
45	132.22	181	1527	26010.73	30856	6270453	10	9	1
46	135.92	182	1564	27169.27	30894	6569019	11	9	1
47	137.26	183	1594	27421.63	31626	6869850	11	8	1
48	134.83	181	1584	28251.49	32452	7129849	11	9	1
49	130.83	178	1551	29071.14	32069	7380800	11	9	1
50	132.28	180	1583	29221.93	32033	7706246	11	9	1
51	133.18	180	1609	30982.74	33021	8024619	11	9	1
52	134.12	183	1613	31925.64	33755	8378238	11	9	1

APÊNDICE B – ALGORITMO GENÉTICO E TRUCK FACTOR

```
import numpy as np
import random
import time
from deap import base
from deap import creator
from deap import tools
from numpy.random import seed
from numpy.random import randint
import time
import logging
import copy

def analyzing_repository_doa(repository):
    np_rep = np.array(repository)
    doa = np.zeros(dtype=int, shape=(np_rep.shape[0], np_rep.shape[1]))
    index = 0
    for i in np_rep:
        doa[index, np.argmax(i, axis=0)] = 1
        index += 1
    return True

def calculate_doa(repository):
    doa = np.zeros(dtype=int, shape=(repository.shape[0],
        repository.shape[1]))
    index = 0
    for i in repository:
        doa[index, np.argmax(i, axis=0)] = 1
        index += 1
    return doa

def removeTopOfAuthors(a):
    authors_contributions = np.sum(a, axis=0)
```

```
    author = np.argmax(authors_contributions, axis=0)
    a = np.delete(a, author, 1)
    return a

def getCoverage(n_files, authors_mapped):
    return np.sum(authors_mapped) / n_files

def calculate_truck_factor(rep_mapped, n_files, n_developers):
    files_size = n_files
    tf = 0
    for i in range(n_developers):
        coverage = getCoverage(files_size, rep_mapped)
        if coverage < 0.5:
            break
        rep_mapped = removeTopOfAuthors(rep_mapped)
        tf += 1
    return tf

def start_tf(repository):
    repository = np.array(repository)
    rep_mapped = calculate_doa(repository)
    return calculate_truck_factor(rep_mapped, repository.shape[0],
        repository.shape[1])

def chromosome():
    return random.choices(range(0, len(np_agents_table)),
        k=len(np_task_table))

def evaluate(individual):
    tmp_repository = copy.copy(np_repository)
    individual = individual[0]
    for i in range(len(individual)):
        agent = individual[i]
        files = list (np_task_file_table[i])
```

```
        for j in range(len(files)):
            if files[j] != -1:
                tmp_repository[files[j]][agent] =
                    tmp_repository[files[j]][agent] +
                    np_task_change_table[i][j]
    truck_factor = start_tf(tmp_repository)
    variance_total = 0
    for i in tmp_repository:
        variance_total += np.var(i)
    return [truck_factor, variance_total]

def find_best_individual(toolbox):
    pop = toolbox.population(n=100)
    fitnesses = list(map(toolbox.evaluate, pop))
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit
    CXPB, MUTPB = 0.2, 0.2
    fits = [ind.fitness.values[0] for ind in pop]
    g = 0
    while g < 50 :
        g = g + 1
        offspring = toolbox.select(pop, len(pop))
        offspring = list(map(toolbox.clone, offspring))
        for child1, child2 in zip(offspring[::2], offspring[1::2]):
            if random.random() < CXPB:
                toolbox.mate(child1[0], child2[0])
                del child1.fitness.values
                del child2.fitness.values
        for mutant in offspring:
            if random.random() < MUTPB:
                toolbox.mutate(mutant[0])
                del mutant.fitness.values

        invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
```

```
        fitnesses = map(toolbox.evaluate, invalid_ind)
        for ind, fit in zip(invalid_ind, fitnesses):
           ind.fitness.values = fit
        pop[:] = offspring
        fits = [ind.fitness.values[0] for ind in pop]
        length = len(pop)
        mean = sum(fits) / length
        sum2 = sum(x * x for x in fits)
        std = abs(sum2 / length - mean ** 2) ** 0.5
        best = pop[custom_max([toolbox.evaluate(x) for x in pop])]
    best = pop[custom_max([toolbox.evaluate(x) for x in pop])]

    return best

def best_agent_skill():
    individual = []
    for i in range(len(np_task_table)):
        task = list(np_task_table[i])
        task_level_required = task[0]
        knowledge_skill = task[1]
        best_agent_index = np_agents_table[:, knowledge_skill].argmax()
        np_agents_table[best_agent_index, knowledge_skill] +=
            task_level_required
        individual.append(best_agent_index)
    return individual

def random_agent():
    values = randint(0, np_agents_table.shape[0], np_task_table.shape[0])
    logging.warn(fBest of Generation {evaluate5([values])})
    return values


def mean_agent_skill():
```

```
    task_agent_number = round(np_task_table.shape[0] /
        np_agents_table.shape[0])
    if task_agent_number == 0:
        task_agent_number = 1
    agents_tmp = []
    values = []
    mean_agents = []
    for i in range(len(np_agents_table)):
        mean_agents.append(round(np_agents_table[i, :].mean()))
    for i in range(np_agents_table.shape[0]):
        for j in range(task_agent_number):
            agents_tmp.append(np.sort(mean_agents)[-1])
        mean_agents.remove(np.sort(mean_agents)[-1])
    return agents_tmp

def custom_max(pop):
    local = np.array(pop)
    max_value = np.argmax(local[:,0])
    min_value = np.argmin(local[:,1])
    if local[max_value,:][0] > local[min_value,:][0]:
        return max_value
    else:
        return min_value

def evaluate_repository(repository, approach):
    total = []
    for i in np.array(repository):
        total.append(round(np.var(i), 2))
    return round(np.mean(total), 2)

def main(repository, a_table, t_table, type, file_table, change_table):
    global np_agents_table
    global np_task_table
    global np_task_file_table
```

```
global np_task_change_table
global np_repository

np_agents_table = np.array(a_table)
np_task_table = np.array(t_table)
np_task_file_table = np.array(file_table)
np_task_change_table = np.array(change_table)
np_repository = np.array(repository)

if type == data/random.txt:
    return random_agent()

if type == data/best.txt:
    individual = best_agent_skill()
    return individual

if type == data/mean.txt:
    individual = mean_agent_skill()
    return individual

creator.create("FitnessMax", base.Fitness, weights=(1.0,-1.0))
creator.create("Individual", list, fitness=creator.FitnessMax)
toolbox = base.Toolbox()
toolbox.register("chromosome", chromosome)
toolbox.register("individual", tools.initRepeat, creator.Individual,
    toolbox.chromosome, n=1)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
best_solution = find_best_individual(toolbox)

return best_solution[0]
```

APÊNDICE C – SCRIPT EM R PARA TESTES ESTATÍSTICOS

```
# Teste de Normalidade
ks_output <- paste0(getwd(), "results/", filename, /ks_, filename, .txt)
hipt_best <- ks.test(case_1$tf_ga, "pnorm", mean = mean(case_1$tf_ga), sd =
    sd(case_1$tf_ga))
capture.output(hipt_best, file = ks_output)
hipt_best <- ks.test(case_1$tf_best, "pnorm", mean = mean(case_1$tf_best),
    sd = sd(case_1$tf_best))
capture.output(hipt_best, file = ks_output, append = TRUE)
hipt_best <- ks.test(case_1$tf_random, "pnorm", mean =
    mean(case_1$tf_random), sd = sd(case_1$tf_random))
capture.output(hipt_best, file = ks_output, append = TRUE)
# Teste de Shapiro
shapiro_output <- paste0(getwd(), "results/", filename, /shapiro_, filename,
    .txt)
shapiro_best <- shapiro.test(case_1$tf_ga)
capture.output(shapiro_best, file = shapiro_output)
# Teste de Wilcox
wilcox_output <- paste0(getwd(), "results/", filename, /wilcox_, filename,
    .txt)
wilcox_test <- wilcox.test(case_1$tf_ga, case_1$tf_random, alternative =
    c("greater"))
capture.output(wilcox_test, file = wilcox_output)
wilcox_test <- wilcox.test(case_1$tf_ga, case_1$tf_random, alternative =
    c("less"))
capture.output(wilcox_test, file = wilcox_output, append = TRUE)
wilcox_test <- wilcox.test(case_1$tf_ga, case_1$tf_random, alternative =
    c("two.sided"))
capture.output(wilcox_test, file = wilcox_output, append = TRUE)
```