UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Uso de NSGA-II e SPACE para composição otimizada de times de desenvolvimento

Bruce Lucien Santos Notario

Monografia - MBA em Ciências de Dados (CEMEAI)

ICMC USP
SÃO CARLOS

Bruce Lucien Santos Notario

Uso de NSGA-II e SPACE para composição otimizada de times de desenvolvimento

Monografia apresentada ao Centro de Ciências Matemáticas Aplicadas à Indústria do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Ciências de Dados.

Área de concentração: Ciências de Dados

Orientadora: Profa. Dra. Cibele Maria Russo Novelli

Versão original

São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Santos Notario, Bruce Lucien
S237u Uso de NSGA-II e SPACE para composição otimizada de times de desenvolvimento / Bruce Lucien Santos Notario; orientadora Cibele Maria Russo Novelli. -- São Carlos, 2024.
86 p.

Trabalho de conclusão de curso (MBA em Ciência de Dados) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. times de desenvolvimento de software. 2. produtividade. 3. multiobjetivo. 4. otimização. 5. agrupamento. I. Russo Novelli, Cibele Maria, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

AGRADECIMENTOS

À minha mãe e ao meu pai. Sem eles eu não seria nada.

À minha esposa. A poesia da minha vida. Simplesmente por existir, mas também por seu apoio, não só nesse trabalho mas em todo o pedaço de espaço-tempo que estamos saboreando juntos.

À minha orientadora. Uma das pessoas mais inteligentes que tive oportunidade de conhecer. Por sua sabedoria e pelas contribuições sempre decisivas.

Ao Banrisul, por todo apoio. Em especial: Carlos Santos (meu gerente, grande apoiador deste trabalho), João Markoski (meu coordenador, por todo auxílio no Projeto de Pesquisa) e a todos os componentes da equipe de Emissão de Cartões de Crédito, da qual tenho a felicidade de fazer parte (um grupo qualificado, com pessoas divergentes em crenças, visões de mundo, valores e experiências, cujas conversas e discussões oferecem um ambiente ímpar para troca e desenvolvimento de ideias).

"Sem medo de altura, é impossível reconhecer a beleza de lugares altos." (LIU, 2017, p. 15)

RESUMO

NOTARIO, Bruce L. S. **Uso de NSGA-II e SPACE para composição otimizada de times de desenvolvimento**. 2024. 86p. Monografia (MBA em Ciências de Dados) - Centro de Ciências Matemáticas Aplicadas à Indústria, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Como indicar o time ideal para cada novo funcionário? O trabalho propõe um método para alocação otimizada de novos desenvolvedores de software, contratados via concurso público, segundo características das equipes existentes em um banco. Foi realizada a análise do problema em contexto real, uma revisão bibliográfica (sobre otimização multiobjetivo) e a montagem de uma situação hipotética do problema concreto, o que permitiu a execução simulada do método. O método foi desenvolvido com base nas estimativas de produtividade geradas por indicadores SPACE e fornece um conjunto ótimo de soluções (uma fronteira de Pareto, obtida com o uso do algoritmo evolutivo NSGA-II). O estudo leva à conclusão de que o método desenvolvido, apesar de eficiente, exige participação humana na definição de indicadores e na avaliação subjetiva de cada uma das soluções encontradas. Sugere-se oportunidades de estudo sobre quais seriam bons indicadores SPACE e como as soluções podem ser utilizadas de forma prática.

Palavras-chave: times de desenvolvimento de software. produtividade. multiobjetivo. otimização. agrupamento.

ABSTRACT

NOTARIO, Bruce L. S. **Use of NSGA-II and SPACE for optimized development team composition**. 2024. 86p. Monograph (MBA in Data Sciences) - Centro de Ciências Matemáticas Aplicadas à Indústria, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

How to indicate the ideal team for each new employee? This study proposes a method for the optimal allocation of new software developers, hired by public competition, according to the characteristics of the existing teams in a bank. The problem was analyzed in a real context, a literature review (on multi-objective optimization), and a hypothetical situation of the concrete problem was created, which allowed the simulated execution of the method. The method was developed based on productivity estimates generated by SPACE indicators and provides an optimal set of solutions (a Pareto front, obtained using the NSGA-II evolutionary algorithm). The study leads to the conclusion that the method developed, despite being efficient, requires human participation in the definition of indicators and the subjective evaluation of each of the solutions found. Opportunities for studying what would be good SPACE indicators and how the solutions can be used practically are suggested.

Keywords: software development teams. productivity. multi-objective. optimization. clustering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
Banrisul	Banco do Estado do Rio Grande do Sul
CSV	*Comma-Separated Values*
JSON	*JavaScript Object Notation*
MOEA/D	*Multi-Objective Evolutionary Algorithm Based on Decomposition*
MOPSO	*Multi-objective Particle Swarm Optimization*
NSGA	*Non-dominated Sorting Genetic Algorithm*
NSGA-II	*Non-dominated Sorting Genetic Algorithm-II*
QA	*Quality Assurance*
POM	Problema de Otimização Multiobjetivo
SPACE	Framework de avaliação de produtividade proposto por Forsgren *et al.* (2021). É a abreviatura de *Satisfaction and Well-Being*, *Performance*, *Activity*, *Communication and Collaboration* e *Efficiency and Flow*.
LGPD	Lei Geral de Proteção de Dados
TCLE	Termo de Consentimento Livre e Esclarecido
REST	*Representational State Transfer*
TI	Tecnologia da Informação
UDS	Unidade de Desenvolvimento de Sistemas

SUMÁRIO

1 INTRODUÇÃO

> (...) Mas as empresas mais valiosas no futuro não perguntarão quais problemas podem ser solucionados pelos computadores sozinhos. Pelo contrário, perguntarão: *Como os computadores podem ajudar os humanos a resolverem problemas difíceis?*(THIEL, 2014, p. 159)

Como empresa pública, o Banco do Estado do Rio Grande do Sul (Banrisul) não pode ir a mercado e contatar diretamente desenvolvedores de software cujo perfil seja de seu interesse: **o banco depende da realização de concursos públicos**. Devido a isso, uma grande variedade de profissionais capacitados acaba por ser admitida nesses processos seletivos. A título de exemplo, no último concurso público promovido pela instituição, foram abertas 124 vagas somente para a área de *Desenvolvimento de Sistemas*, o que corresponde a 45% do total de vagas oferecidas para o cargo de *Técnico em Tecnologia da Informação*, fato que pode ser observado na Tabela 1.

Tabela 1 – Vagas por área no concurso público de 2022 do Banrisul para o cargo de Técnico em Tecnologia da Informação.

		Vagas		
Área	**Descrição**	**Quantidade**	**%**	**Ilustração**
1	Analista de Segurança da TI	22	8	||
2	Analista de Transformação Digital	14	5	|
3	**Desenvolvimento de Sistemas**	**124**	**45**	||||||||
4	Gestão de TI	59	22	||||
5	*Quality Assurance* (QA)/Analistas de Teste	13	5	|
6	Suporte à Infraestrutura de TI	32	12	||
7	Suporte à Plataforma Mainframe	10	4	|
	Total:	274	100	|||||||||||||||||||

Fonte: adaptado de Banrisul (2022, p. 6).

Decidir como alocar aprovados em concursos públicos é um detalhe importante para o sucesso dos projetos sob responsabilidade da Unidade de Desenvolvimento de Sistemas (UDS) do Banco. É uma decisão que precisa considerar sim a carência de profissionais em determinados setores e times, mas também exige reflexão sobre quem são os novos funcionários: nível e perfil técnico, experiências profissionais anteriores, capacidade de colaboração e de comunicação, o quanto estão satisfeitos em trabalhar para o Banrisul. Exemplos da complexa subjetividade inerente às relações humanas, que se manifesta também na alocação de profissionais.

Mas até que ponto a alocação de candidatos aprovados em concursos públicos do Banrisul está alinhada com os conceitos mais modernos de **montagem de times** e de

avaliação de produtividade? E se fosse possível minimizar os efeitos subjetivos com a recomendação automatizada do time ideal para determinado candidato? Um sistema de recomendação que foque em otimizar a produção da empresa com a utilização de indicadores realmente significativos, analisando o potencial do novo funcionário sem esquecer que ele é humano e que, por isso, tem sua subjetividade? Eis aqui o objetivo principal do projeto de pesquisa agora proposto.

1.1 Problema de pesquisa

Como criar modelos de aprendizado de máquina que indiquem o time ideal para o qual um desenvolvedor aprovado em concurso público deve ser alocado?

1.2 Objetivo geral

- Automatizar o processo de escolha do time para o qual será alocado um novo funcionário.

1.3 Objetivos específicos

- Oferecer aos gestores da UDS uma forma fácil e assertiva de avaliar qual é o time ideal para o qual um desenvolvedor aprovado em concurso público deve ser alocado;
- Desenvolver um modelo de recomendação que vise otimizar a produtividade do novo desenvolvedor, do time no qual será alocado e da UDS como um todo;
- Aplicar na UDS os conceitos do *framework* de avaliação de produtividade denominado SPACE (FORSGREN *et al.*, 2021).

2 REFERENCIAL TEÓRICO

Os objetivos específicos (Seção 1.3) serviram de referência para o início da elaboração do referencial teórico. Sabia-se antecipadamente, portanto, que (1) **o *Framework* SPACE seria utilizado** para estimação de produtividade. Era claro também que (2) **a produtividade deveria ser otimizada** em 3 níveis de granularidade: *UDS*, *times* e *desenvolvedores*. Outra característica inicial do estudo era (3) o **foco no ponto de vista dos tomadores de decisão**.

2.1 SPACE

Conforme proposto por Forsgren *et al.* (2021), a produtividade de desenvolvedores de software não pode ser medida por apenas um indicador e, devido a isso, os autores apresentam 5 dimensões (ou *características*) a serem estimadas quantitativamente: "Satisfação e Bem-Estar", "Performance", "Atividade", "Comunicação e Colaboração" e "Eficiência e Fluxo"[1]. Estas dimensões formam o *framework* SPACE. Todas podem ser aplicadas a diferentes níveis hierárquicos e, ainda conforme os autores, um mínimo de 3 delas são necessárias para que se avalie a produtividade de forma significativa. Segue abaixo uma breve descrição do que se trata cada uma dessas dimensões (FORSGREN *et al.*, 2021).

- **Satisfação e Bem-Estar**. *Satisfação* é o quanto os desenvolvedores de software se sentem satisfeitos com seu trabalho, time, ferramentas ou cultura; *Bem-Estar* é o quão saudáveis e felizes esses desenvolvedores estão e quanto o trabalho impacta nisso; exemplos de indicadores: *grau de satisfação dos funcionários*, *nível de* burnout (FORSGREN *et al.*, 2021).

- **Performance** é o resultado final, o impacto causado por um sistema ou processo. É a resposta para a pergunta *o código escrito pelo desenvolvedor faz o que se supõe que ele deveria fazer?*; exemplos de indicadores: *nível de satisfação do cliente em relação ao software*, *quantidade de bugs* (FORSGREN *et al.*, 2021).

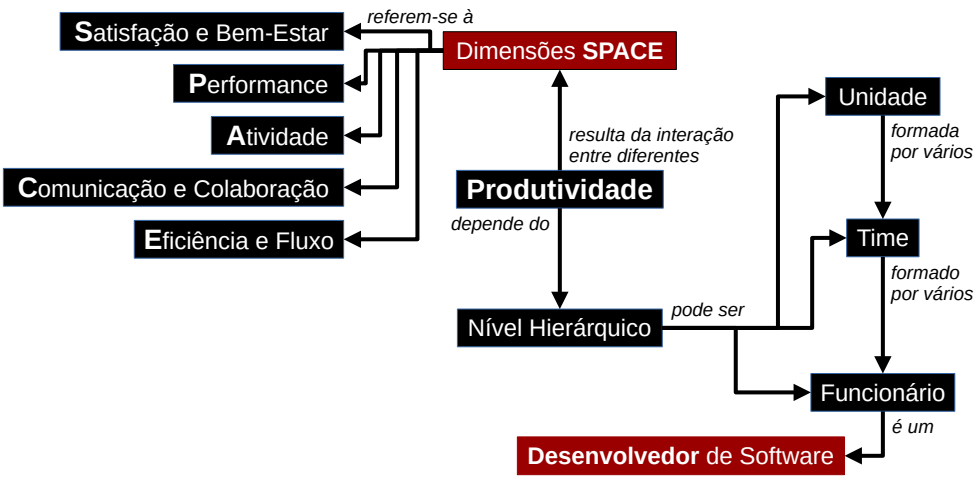
- **Atividade** é a contagem de ações ou saídas durante a execução do trabalho; exemplos de indicadores: *quantidade de linhas de código produzidas*, *quantidade de commits* (FORSGREN *et al.*, 2021).

[1] Termos originais: *Satisfaction and Well-Being*, *Performance*, *Activity*, *Communication and Collaboration* e *Efficiency and Flow*. Tradução nossa.

- **Comunicação e Colaboração** refere-se a como pessoas e times se comunicam e trabalham juntos. exemplos de indicadores: *velocidade de integração de código*, *quem se comunica com quem* (FORSGREN *et al.*, 2021).
- **Eficiência e Fluxo**, relacionada com todas as outras dimensões, é a habilidade de completar ou fazer progressos no trabalho com o mínimo de interrupções ou atrasos, seja individualmente ou através de um sistema. exemplos de indicadores: *quantidade de interrupções*, *tempo total para realizar uma tarefa* (FORSGREN *et al.*, 2021).

Na Figura 1 pode-se visualizar um mapa conceitual que relaciona as dimensões SPACE, o conceito de produtividade e os diferentes níveis hierárquicos considerados neste estudo. Produtividade aqui é o resultado da interação entre as diferentes dimensões SPACE, que podem ser aplicadas isoladamente a desenvolvedores de software, a times de desenvolvimento de software (compostos por vários desenvolvedores) ou unidades de desenvolvimento de software (formadas por vários times).

Figura 1 – Conceitos básicos de produtividade utilizados no trabalho.



Fonte: elaborado pelo autor tendo como referência os conceitos apresentados por Forsgren *et al.* (2021).

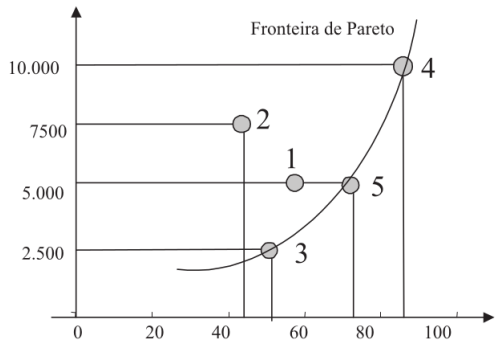
A forma como Forsgren *et al.* (2021) apresentam o *framework* SPACE deixa clara a interdependência entre as dimensões. Contudo, essas dimensões são tratadas como se fossem independentes sob o ponto de vista da importância. Essa é a situação típica de problemas nos quais há um *trade-off* claro entre escolhas. O aumento no valor estimado para uma dimensão pode levar a uma diminuição no valor estimado em outra, e vice-versa.

Esse *trade-off* não raras vezes implica em duas opções aceitáveis, mas que exigem uma escolha (frequentemente subjetiva) de qual delas será priorizada. É por isso que a resposta para um processo de otimização de dimensões SPACE é ideal para a busca não de uma, mas de várias soluções possíveis. Ao conjunto de soluções possíveis para um processo de otimização que envolve *trade-offs* atribui-se o nome **fronteira de Pareto** (GARSHASBI *et al.*, 2019, p. 508-509).

2.2 Fronteira de Pareto

A Figura 2 é apresentada ao leitor com o objetivo de esclarecer o que é uma fronteira de Pareto. Nela, os pontos 1, 2, 3, 4 e 5 são soluções possíveis para um problema com apenas duas dimensões concorrentes mas inter-relacionadas, representadas pelos eixos do gráfico. As escalas desses eixos são propositalmente diferentes (TICONA WALDO GONZALO CANCINO E DELBéM, 2008).

Figura 2 – Exemplo de fronteira de Pareto.



Fonte: adaptada de Ticona Waldo Gonzalo Cancino e Delbém (2008, p. 6).

A fronteira de Pareto propriamente dita é composta pelos pontos 3, 4 e 5. Diz-se que, no eixo horizontal, o ponto 1 é *dominado* pelos pontos 4 e 5, a despeito do ponto 1 dominar o ponto 3 nesse mesmo eixo. Essa lógica de *dominância*, quando aplicada nos dois eixos, conduz à percepção de que há soluções melhores, que *dominam* outras soluções, como é o caso das soluções 3, 4 e 5. A Figura 2 mostra um exemplo em duas dimensões, mas o conceito é extensível a N dimensões (TICONA WALDO GONZALO CANCINO E DELBéM, 2008).

Outrossim, cabe destacar aqui que, usualmente, uma fronteira de Pareto é o resultado esperado para toda uma classe de problemas conhecida como **Otimização Multiobjetivo** (GARSHASBI *et al.*, 2019, p. 508-509).

2.3 Otimização Multiobjetivo

É comum que uma única solução seja encontrada para problemas nos quais tenciona-se otimizar o valor de apenas uma variável. No entanto, quando há mais de uma variável envolvida, pode-se executar um processo de otimização que vise otimizar todas elas de forma simultânea. O resultado esperado é um conjunto de **soluções na forma de vetores**, que normalmente é entendido como uma fronteira de Pareto. Essas soluções representam *trade-offs* no espaço e decisão do problema. Ao final do processo, o tomador de decisão precisa escolher uma dessas soluções contidas na fronteira de Pareto encontrada. Simplificadamente, ao escolher uma solução, o tomador de decisão escolhe um vetor entre os vários vetores potenciais encontrados pelo processo de otimização (VELDHUIZEN; LAMONT *et al.*, 1998).

Problemas de otimização que envolvem $N > 1$ variáveis e cujo resultado é uma fronteira de Pareto são chamados de *Problemas de Otimização Multiobjetivo* (POMs). Busca-se na resolução desses problemas *minimizar*, de forma simultânea, o valor de todas as variáveis envolvidas no problema; caso o problema seja de de *maximização* então converte-se ele para *minimização* (VELDHUIZEN; LAMONT *et al.*, 1998).

Um POM pode ser definido como (VELDHUIZEN; LAMONT *et al.*, 1998):

$$\begin{aligned} &\text{minimizar} \quad f(x) = (f_1(x), ..., f_p(x)) \\ &\text{sujeito a} \quad g_i(x) \leq 0, i = 1, ..., m \end{aligned} \tag{2.1}$$

Sendo n o número de variáveis do problema, na definição acima x é uma variável de decisão na forma de um vetor n-dimensional, m é o número de restrições e p é a quantidade de objetivos, que normalmente são conflitantes entre si (VELDHUIZEN; LAMONT *et al.*, 1998).

Nesse ponto do estudo sabia-se que os indicadores de produtividade dos desenvolvedores seriam estimados com base em SPACE (Seção 2.1), que tais indicadores seriam utilizados como entrada de um POM e que os gestores poderiam escolher uma solução adequada entre as soluções da fronteira de Pareto (Seção 2.2) encontrada após a resolução do POM. O próximo passo seria encontrar uma forma de chegar à fronteira de Pareto desejada, ou seja, entender como o POM poderia ser resolvido. Essa foi a motivação para a realização de uma **revisão bibliográfica**.

2.4 Revisão bibliográfica

2.4.1 Questão de pesquisa

Como a Academia tem abordado a resolução de problemas de otimização multiobjetivo em contextos colaborativos?

2.4.2 *String* de busca

Para responder a questão de pesquisa definida para a revisão bibliográfica, optou-se por elaborar uma *string* de busca composta por três partes principais: (1) colaboração e montagem de equipes, (2) otimização e (3) estratégia multiobjetivo. Dessa forma, a *string* de busca final foi definida como "**(team composition OR team effectiveness OR group effectiveness OR team building OR teamwork) AND optimization AND multi-objective**".

2.4.3 Fonte e estratégia de busca

Adotou-se uma fonte de busca abrangente: o **Portal de Periódicos CAPES**; que foi utilizado com a seguinte *estratégia de busca*:

- Acessar http://periodicos.capes.gov.br;
- Realizar "Acesso CAFe" com "login USP";
- Realizar busca utilizando a *string de busca* definida anteriormente (Seção 2.4.2);
- Refinar busca restringindo:
  - "Data de Criação" entre 2013 e 2023;
  - "Disponibilidade" como:
    - "Periódicos revisados por pares"; e
    - "Recurso On-line".
  - "Tipo de recurso" como "Artigos".
- Certificar-se de que "Ordenar por" está marcado como "Relevância";
- Realizar o download do conteúdo (PDF) dos primeiros 10 resultados.

2.4.4 Seleção de estudos

Os **critérios de inclusão** foram:

- Ano de publicação está dentro do período de 2013 a 2023:
  - Busca foi realizada em 08/06/2023.
- Foi revisado por pares;
- Conteúdo está disponível na internet;
- É publicação do tipo "Artigo".

Os **critérios de exclusão** foram:

- Conteúdo está inacessível devido a restrições de acesso ou endereço da fonte indisponível:
  - Eliminou (1) Azouz and Boughaci (2023) e (2) Hosseini and Akhavan (2017).
- Não é um dos 10 primeiros resultados da busca realizada:
  - Total de 351 resultados foi retornado na busca.
- Não resolve um problema prático, relacionado com situações reais:
  - Eliminou Guo, Tang and Niu (2022).
- Conteúdo não foi escrito na língua inglesa.

2.4.5 Síntese dos dados

Foram analisados, ao todo, 7 estudos. Os títulos desses estudos podem ser observados na Tabela 2.

Tabela 2 – Títulos originais dos estudos analisados.

Trabalho	**Título**
Si *et al.* (2022)	*A reliability-and-cost-based framework to optimize maintenance planning and diverse-skilled technician routing for geographically distributed systems*
Zhang, Ma and Jiang (2022)	*Research on Multi-Objective Multi-Robot Task Allocation by Lin–Kernighan–Helsgaun Guided Evolutionary Algorithms*
Zhao *et al.* (2021)	*Multi-Objective Optimization for Football Team Member Selection*
Garshasbi *et al.* (2019)	*Optimal learning group formation: A multi-objective heuristic search strategy for enhancing inter-group homogeneity and intra-group heterogeneity*
Shukla *et al.* (2018)	*Multi-objective cross-version defect prediction*
Shao, Geyer and Lang (2014)	*Integrating requirement analysis and multi-objective optimization for office building energy retrofit strategies*
Zhang and Zhang (2013)	*Multi-objective team formation optimization for new product development*

Fonte: elaborado pelo autor.

Os estudos analisados não se restringiram a uma única revista ou editora científica, como pode ser observado na Tabela 3. Além disso, se destaca o fato de que os artigos analisados foram publicados em veículos dedicados a temas relativamente sortidos, como engenharia, computação, matemática, indústria e energia.

Tabela 3 – Editoras e revistas associadas aos estudos analisados.

Trabalho	**Editora**	**Revista**
Si *et al.* (2022)	Elsevier	*Reliability Engineering and System Safety*
Zhang, Ma and Jiang (2022)	MDPI	*Mathematics*
Zhao *et al.* (2021)	IEEE	*IEEE Access*
Garshasbi *et al.* (2019)	Elsevier	*Expert Systems With Applications*
Shukla *et al.* (2018)	Springer	*Soft Computing*
Shao, Geyer and Lang (2014)	Elsevier	*Energy and Buildings*
Zhang and Zhang (2013)	Elsevier	*Computers & Industrial Engineering*

Fonte: elaborado pelo autor.

A fim de responder a questão de pesquisa proposta para a revisão bibliográfica (Seção 2.4.1), elaborou-se a Tabela 4. Nela pode-se observar um breve resumo do problema que foi abordado no estudo, a tarefa de otimização que foi proposta e uma lista dos algoritmos utilizados pelos pesquisadores.

Tabela 4 – Problema, tarefa e algoritmos associados aos estudos analisados.

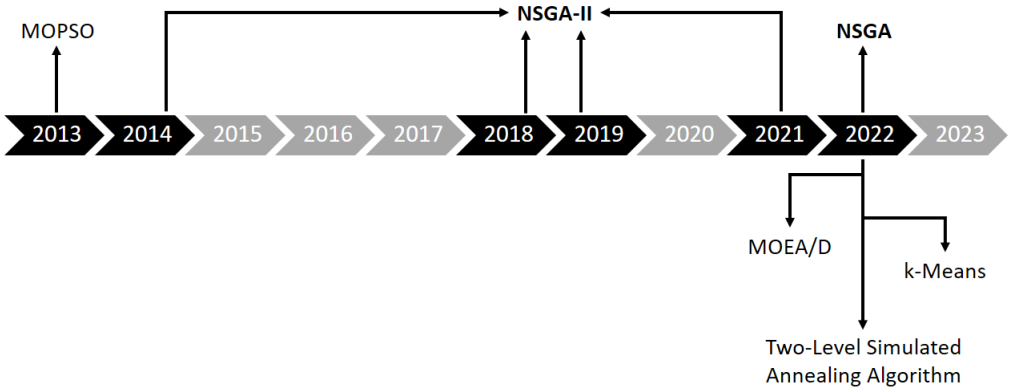
Trabalho	**Problema**	**Tarefa**	**Algoritmos**
Si *et al.* (2022)	Alocação de técnicos em tarefas de manutenção preventiva e de reposição em máquinas.	**maximizar** confiabilidade das máquinas; **minimizar** custo do serviço prestado	*k-Means*; *Two-Level Simulated Annealing Algorithm*
Zhang, Ma and Jiang (2022)	Alocação de tarefas entre múltiplos robôs.	**minimizar** soma das distâncias percorridas pelos robôs; **minimizar** distância máxima percorrida por cada robô	NSGA; MOEA/D
Zhao *et al.* (2021)	Escolha de membros de times de futebol.	**maximizar**, isoladamente, a soma de cada uma de 5 características dos jogadores	NSGA-II
Garshasbi *et al.* (2019)	Agrupamento de alunos visando aprendizado colaborativo.	**maximizar** homogeneidade na comparação entre grupos; **maximizar** heterogeneidade dos alunos dentro de cada grupo no qual são colocados	NSGA-II
Shukla *et al.* (2018)	Predição de arquivos fonte de software propensos a defeitos.	(Abordagem 1) **maximizar** *recall*; **minimizar** custo de correção de defeitos – (Abordagem 2) **Maximizar** *recall*; **minimizar** LOC[3]	NSGA-II
Shao, Geyer and Lang (2014)	Projeto de atualização energética (*energy retrofit*) de uma construção do ano de 1900.	**minimizar** custo do investimento inicial; **minimizar** gasto operacional anual de energia; **minimizar** emissões equivalentes de CO^2.	NSGA-II
Zhang and Zhang (2013)	Formação de um time de desenvolvimento de produto.	**maximizar** soma das estimativas das capacidades exigidas pelo time; **maximizar** quão bem relacionados são os funcionários dentro e fora de seus departamentos	MOPSO

Fonte: elaborado pelo autor.

2.4.6 Resultados da revisão bibliográfica

Nos últimos 10 anos a Academia tem abordado problemas de otimização multiobjetivo com o uso de heurísticas baseadas na observação da natureza. Os algoritmos utilizados nos estudos analisados foram (Figura 3): *Two-Level Simulated Annealing Algorithm* (crescimento de cristais), MOPSO (enxames de animais), MOEA/D (teoria da evolução) e NSGA (também baseado na teoria da evolução). A única exceção foi o uso de *k-Means* em Si *et al.* (2022); o estudo que, outrossim, faz uso de MOPSO (Tabela 4).

Figura 3 – Linha do tempo dos algoritmos utilizados nos estudos analisados.



Fonte: elaborado pelo autor com base nos dados da Tabela 4.

Conforme os dados expostos na Seção 2.4.5, a abordagem mais popular entre os pesquisadores parece ser a evolucionista. Mais especificamente, o uso do algoritmo conhecido como NSGA-II. Nesse algoritmo procura-se o que se chama de fronteira de Pareto, um conjunto de soluções ótimas que satisfazem tanto o problema de otimização quanto as restrições impostas a ele (ver Seção 2.2).

Garshasbi *et al.* (2019) se sobressai entre os artigos analisados por sua estratégia para definição de grupos de estudantes. Os autores do estudo definiram os cromossomos como sendo compostos por sequências de estudantes (e não como sequências de zeros e uns como é a abordagem clássica de algoritmos genéticos). Outro detalhe digno de comentário é a definição dos grupos de estudantes como divisões do próprio cromossomo, o que implicou na adaptação da função de aptidão utilizada no estudo.

Por fim, provavelmente devido ao filtro aplicado na *string* de busca (Seção 2.4.2), pode-se notar que mais da metade dos estudos analisados resolvem problemas de agrupamento otimizado de pessoas com a finalidade de otimizar a realização de tarefas colaborativas. Fazem parte desses estudos Si *et al.* (2022), Zhao *et al.* (2021), Garshasbi *et al.* (2019) e Zhang and Zhang (2013).

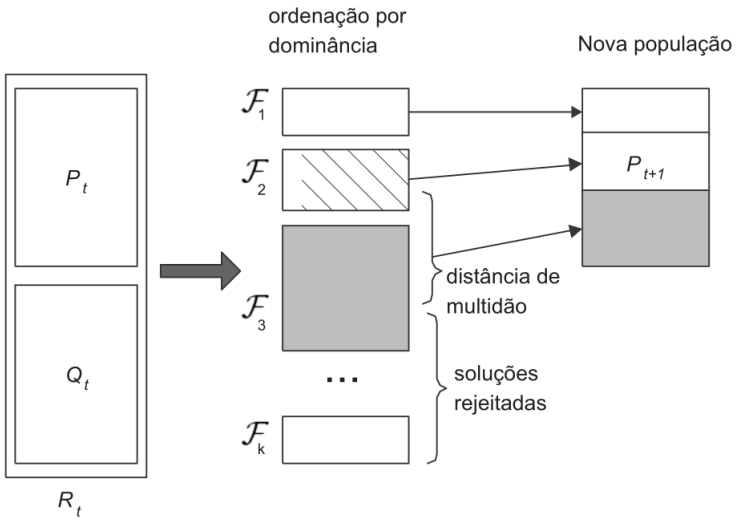
2.5 NSGA-II

NSGA-II foi o algoritmo citado com mais frequência na revisão bibliográfica (Seção 2.4.6). Assim sendo, essa abordagem foi a forma de resolução escolhida para o POM identificado neste estudo e, por isso, será detalhado aqui.

Proposto por Deb *et al.* (2002), **NSGA-II** foi desenvolvido para resolução de problemas multiobjetivo. Explicando simplificadamente, NSGA-II usa **dois conjuntos de soluções** em cada iteração (ou *geração*). Tais conjuntos são usualmente chamados de P e Q. Na primeira geração, P e Q estão vazios, o que exige geração aleatória de soluções para o conjunto P. Às soluções em P são aplicados *operadores genéticos* (de *seleção*, *crossover* e *mutação*) para geração das soluções que farão parte do conjunto Q. A população resultante da operação de *união* dos conjuntos P e Q é então ordenada por **dominância** e **diversidade** e, dessa mesma população formada por $P \cup Q$, são retirados os indivíduos que farão parte do grupo P da próxima geração. O processo então se repete.

A Figura 4 ilustra essa descrição. Nessa figura, $R = P \cup Q$ e t indica a *iteração*. Cada $\mathcal{F}$ nessa mesma ilustração refere-se a uma fronteira de Pareto específica encontrada pelo algoritmo no conjunto R, o que classifica as soluções em níveis diferentes de *dominância*. A *distância de multidão* é uma medida utilizada para garantir a *diversidade* entre as soluções.

Figura 4 – Esquema de funcionamento do NSGA-II.

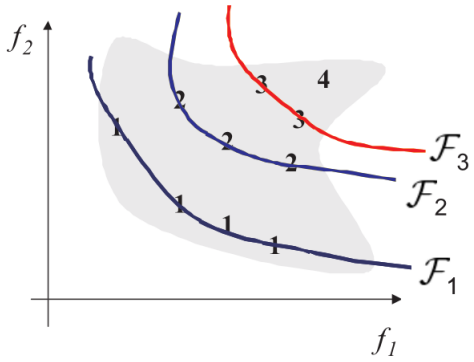


Fonte: Ticona Waldo Gonzalo Cancino e Delbém (2008, p. 23).

Na Figura 5 pode-se ver (em duas dimensões) uma ilustração do que seriam esses níveis de dominância. NSGA-II procura um primeiro nível de soluções dominantes e, na sequência, identifica o segundo nível, e assim por diante, até que não estejam disponíveis mais soluções. O uso de duas dimensões na Figura 5 facilita a compreensão do conceito

referente aos níveis de dominância. No entanto, ele também é extensível a 3 ou mais dimensões.

Figura 5 – Fronteiras de Pareto geradas pelo NSGA-II a fim de classificar as soluções encontradas em níveis de *dominância*.



Fonte: Ticona Waldo Gonzalo Cancino e Delbém (2008, p. 22).

A seleção de soluções em P para geração dos descendentes em Q é realizada por o que é chamado de *torneio*. Um torneio considera, primeiro, se a solução é mais *dominante* do que sua adversária e, em segundo lugar, qual das duas é mais única, de modo a promover a *diversidade* das soluções. A dominância é obtida pela classificação das soluções em *níveis de dominância*. O que, na prática, se materializa em várias fronteiras de Pareto, uma mais dominante do que a outra, mas escolhidas adequadamente dentro do mesmo conjunto de soluções (Figura 5). A *diversidade* é garantida pela maximização da distância de cada solução a suas soluções adjacentes, ou seja, da *distância de multidão*.

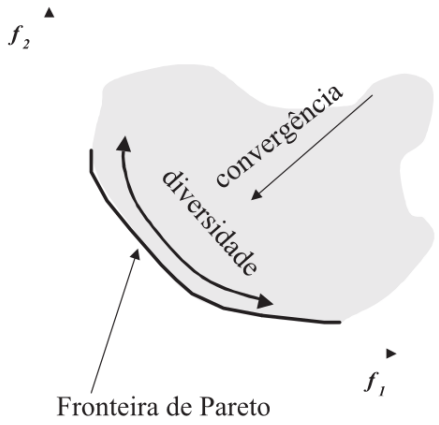
Dominada a lógica envolvida na implementação do **NSGA-II**, restava ainda a escolha de uma métrica que indicasse, de maneira objetiva, como avaliar as soluções encontradas. Cada execução desse algoritmo obtém um conjunto diferente de soluções para um POM e, por isso, é mister o uso de uma métrica para comparação entre os resultados de todas as execuções. A métrica escolhida para avaliar os conjuntos de soluções entre si foi o **hipervolume**.

2.6 Hipervolume

Conforme Ticona Waldo Gonzalo Cancino e Delbém (2008, p. 32), a avaliação de algoritmos multiobjetivo evolucionários, como é o caso do NSGA-II, precisa de dois indicadores diferentes. Um apenas para medir quão bem as soluções convergem para a fronteira de Pareto ideal (**convergência**) e outra para medir quão diversas são as soluções (**diversidade**). Além disso, também segundo esses autores, essas características podem ser conflitantes no

problema abordado. A Figura 6 procura ilustrar os conceitos de convergência e diversidade apresentados aqui.

Figura 6 – Diferença entre os conceitos de *convergência* e de *diversidade*.



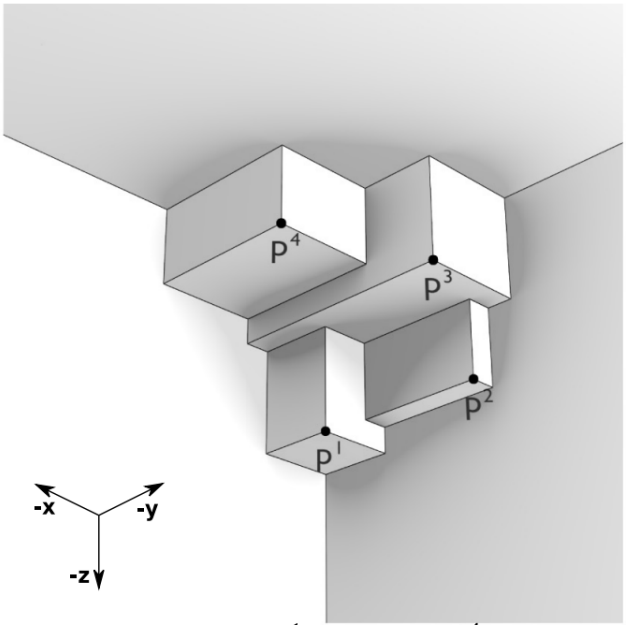
Fonte: Ticona Waldo Gonzalo Cancino e Delbém (2008, p. 31).

Dito isso, busca-se nesse estudo um conjunto diversificado de soluções que representem a melhor aproximação possível para a fronteira de Pareto ideal do problema, que é desconhecida *a priori*. A avaliação da qualidade destas soluções, portanto, precisa tornar visível quão melhor o conjunto de soluções é em relação à *solução base* (termo adotado neste estudo; é aquela solução na qual não são considerados os novos desenvolvedores; comentada mais detalhadamente na Seção 4.4) e indicar como menos relevantes soluções mais concentradas.

Neste trabalho será utilizado um único indicador, para medir tanto convergência quanto diversidade. Este indicador é o **hipervolume**, que associa um cálculo específico de volume ao conjunto de soluções sob avaliação. Em uma dimensão, tal indicador seria a medida de um seguimento de reta; em duas, um cálculo específico de área; em três dimensões, um volume propriamente dito, calculado de forma apropriada. Em $N > 3$ dimensões estende-se esses cálculos apropriados para o que é chamado de *hipervolume*.

Assim sendo, utiliza-se o volume do *poliedro* formado pela mesclagem de todos os *paralelepípedos retângulos retos* cuja origem no *plano cartesiano* é o *vértice* que representa a *solução base* e cujo ponto mais distante da origem é uma das soluções encontradas. Uma ilustração desse tipo de *poliedro* é a Figura 7, na qual os pontos $p^1, ..., p^4$ representam 4 soluções hipotéticas. Uma explicação detalhada sobre hipervolume pode ser encontrada em Guerreiro, Fonseca and Paquete (2021). De forma mais objetiva, *quanto maior o valor do hipervolume calculado a partir de um conjunto de soluções, melhor é este conjunto de soluções.*

Figura 7 – Ilustração do *poliedro* cujo volume é calculado para avaliar a qualidade de um determinado conjunto de soluções $(p^1, ..., p^4)$.



Fonte: Guerreiro, Fonseca and Paquete (2021, p. 5).

A definição de hipervolume conclui o estudo sobre a teoria que suporta a solução do problema proposto. Apesar disso, a fim de agilizar a obtenção dos resultados desejados, minimizar o risco de equívoco na implementação do código do algoritmo e, principalmente, focar na obtenção da fronteira de Pareto desejada, este trabalho fez uso de uma biblioteca de código construída especificamente para resolução de POMs, a biblioteca **pymoo**.

2.7 pymoo

pymoo[4] (Blank; Deb, 2020) é uma biblioteca de código escrita em linguagem *Python* que, além de muitas outras vantagens, oferece implementações prontas de vários algoritmos para resolução de POMs. Entre estes algoritmos está o NSGA-II (Seção 2.5).

A implementação de NSGA-II no pymoo está materializada na classe $NSGA2$[5], cujo construtor oferece possibilidades variadas de configuração do algoritmo. No contexto desse trabalho, sobressaem-se os seguintes parâmetros do construtor:

- pop_size: tamanho da população inicial (hiperparâmetro);
- $n\_offsprings$: quantidade de descendentes gerados em cada iteração (hiperparâmetro);

[4] pymoo.org
[5] Documentação em pymoo.org/algorithms/moo/nsga2.html.

- *sampling*: tipo de amostragem realizada na primeira geração (hiperparâmetro);
- *crossover*: estratégia de *crossover* (*recombinação*) entre pares de soluções;
- *mutation*: estratégia de mutação aplicada a uma solução isolada;
- *eliminate_duplicates*: eliminação de soluções duplicadas.

Os parâmetros *sampling*, *crossover* e *mutation* são particularmente importantes para este estudo pois precisam considerar soluções cujas combinações não repitam desenvolvedores. O problema de otimização abordado aqui é semelhante àquele apresentado por Garshasbi *et al.* (2019), no qual também há a otimização de grupos a partir de determinadas características. A abordagem de Garshasbi *et al.* (2019) considera que cada *cromossomo* é uma solução, o que é exatamente o que será necessário neste trabalho; uma das pré-suposições implícitas nos objetivos geral e específicos é a de que um novo desenvolvedor só pode ser alocado a um único time.

Posto isso, neste estudo, *sampling*, *crossover* e *mutation* precisam, necessariamente, ser informados de forma a seguir uma abordagem de *permutação*[6]. Na prática, os vetores que representam as soluções são preenchidos com números inteiros de 1 a N, onde N é a quantidade de possibilidades do problema. No contexto desse trabalho, cada número inteiro desse vetor representaria um novo desenvolvedor. Em resumo:

- *sampling*: recebe um objeto $PermutationRandomSampling$ pois ele cria aleatoriamente soluções garantindo que cada uma delas não contenha números repetidos;
- *crossover*: recebe um objeto $OrderCrossover$ pois esta classe permite que duas soluções troquem entre si sequências de números com a garantia de que, em cada uma delas, não exista repetição de números;
- *mutation*: recebe a instanciação de $InversionMutation$ a fim de garantir que a alteração aleatória de uma única solução mantenha a solução com números não repetidos.

A classe $NSGA2$ é apenas uma das várias opções possíveis de escolha de algoritmos oferecidos por pymoo. Como os algoritmos são intercambiáveis (até certo limite), também existe uma classe que precisa ser estendida de forma a expressar as particularidades do problema que se deseja solucionar.

Essa classe é chamada de $ElementwiseProblem$ e exige a implementação do método $\_evaluate$, que é o código que retorna um valor estimado de qualidade para uma

[6] Documentação em pymoo.org/customization/permutation.html

determinada solução dada (também denominada *função de aptidão*). Além da implementação de *_evaluate*, *ElementwiseProblem* exige o informe de dois parâmetros em seu construtor. São eles:

- *n_var*: tamanho do vetor que representa uma solução;
- *n_obj*: quantidade de *objetivos* a serem minimizados.

Tanto *NSGA2* quanto *ElementwiseProblem* permitem a criação de instâncias que devem ser repassadas à função *minimize*, que é quem verdadeiramente realiza a procura pelas soluções otimizadas. Os parâmetros dessa função que são relevantes para o estudo em questão são:

- *problem*: instância de *ElementwiseProblem*;
- *algorithm*: instância de *NSGA2*;
- *termination*: critério de término da execução do algoritmo;
- *seed*: número inteiro referente à semente aleatória;
- *save_history*: permite a avaliação posterior dos resultados intermediários do processo de minimização;
- *verbose*: imprime resultados intermediários;

Além de oferecer facilidades para resolução de POMs, pymoo também disponibiliza uma implementação facilitada para o cálculo de hipervolume (Seção 2.6): a classe *Hypervolume*[7]. Os parâmetros relevantes dela para o problema em estudo aqui são:

- *ref_point*: ponto de referência a partir do qual o hipervolume será calculado;
- *norm_ref_point*: informa se os valores passados em *ref_point* precisam ser normalizados;
- *zero_to_one*: informa se o cálculo do hipervolume deve retornar um valor normalizado;
- *ideal*: estimativa do ponto com a solução ideal;
- *nadir*: estimativa do ponto diretamente oposto ao ideal.

[7] Documentação em pymoo.org/misc/indicators.html

A biblioteca pymoo facilitou enormemente a implementação da solução para o problema proposto, mas tudo que foi comentado e citado até este ponto ataca apenas superficialmente o problema que este trabalho se propõe a resolver. Se faz necessário, portanto, o aprofundamento no mundo real, no contexto verdadeiro do problema que motiva o estudo. É por isso que também realizou-se aqui um **estudo do negócio**.

2.8 Estudo do negócio

O estudo do negócio foi marcado por algumas atividades principais: (1) entrevistas informais com gestores da UDS (que revelaram detalhes sobre o processo de *onboarding* de novos desenvolvedores (Seção 2.8.1) além de particularidades sobre como os times são formados (Seção 2.8.2)) e (2) a tentativa de acesso aos dados reais de produtividade dos desenvolvedores (Seção 2.8.3).

2.8.1 Considerações gerais dos gestores

Em 26/05/2023 foram realizadas entrevistas informais com alguns gestores da UDS. Nessas entrevistas, considerações pertinentes foram levantadas, tanto para a realização do estudo quanto para o entendimento do problema proposto. Abaixo, uma lista das principais considerações discutidas com os gestores.

2.8.1.1 Não há exigência de formação em TI

No último concurso realizado pelo Banrisul (edital Banrisul (2022)), entre os aprovados para a área de Desenvolvimento de Sistemas, podem-se encontrar as profissões de *dentista*, *piloto de avião* e *engenheiro civil*. Todos formados em suas áreas originais de atuação. Exigir um diploma relacionado com TI não foi uma exigência do último edital devido à percepção dos gestores de que muitos deles próprios não tem formação na área de desenvolvimento de sistemas (a título de exemplo, há um gestor economista).

Existe, no entanto, um reconhecimento de que, na época em que estes gestores cursavam a graduação, cursos na área de TI eram raros ou, até mesmo, inexistentes. Realidade completamente diferente da atual, na qual há cursos focados em programação e análise de sistemas, como cursos de graduação em Engenharia de Software, por exemplo.

2.8.1.2 Probatório inadequado para necessidades específicas

No último concurso realizado para o preenchimento de vagas na UDS os candidatos passaram por um período de treinamento de três meses. O treinamento foi composto por duas etapas:

- Dois meses de treinamento dedicado apenas ao *framework* mais comum utilizado pelas equipes de desenvolvimento do Banrisul (desenvolvido pelo próprio

Banrisul); e

- Um mês trabalhando diretamente nas equipes para as quais seriam definitivamente alocados.

Salienta-se que ambas as etapas contém avaliações. Uma prova foi aplicada ao final da primeira etapa, consistindo do desenvolvimento de um pequeno sistema usando o *framework* desenvolvido internamente, no Banrisul. A chefia direta na equipe para a qual o desenvolvedor foi alocado realizou a segunda avaliação.

Não é a primeira vez que os três meses de treinamento são oferecidos aos novos desenvolvedores. A novidade, neste último concurso, foi a fase final nas próprias equipes de desenvolvimento, algo que não ocorreu nos concursos anteriores. Esse detalhe foi alterado com o objetivo de atenuar o impacto da entrada do novo desenvolvedor na equipe.

Como o banco trabalha com muitas tecnologias diferentes e o treinamento inicial é dedicado somente a uma única delas, é relativamente comum que os novos desenvolvedores enfrentem problemas posteriores nas equipes. Principalmente quando a equipe para a qual foram alocados não trabalha com o *framework* desenvolvido internamente. Há casos nos quais, por exemplo, por alguma necessidade do banco e aptidão prévia do novo funcionário, novos desenvolvedores são alocados em projetos inteiramente novos, que não tem relação alguma com a manutenção dos sistemas internos.

2.8.1.3 Habilidade complementar não garante aumento de produtividade

Segundo a experiência dos líderes entrevistados, nem sempre observa-se aumento de produtividade no time após a inclusão de uma pessoa com características complementares àquelas predominantes nele. Por exemplo, uma pessoa comunicativa alocada em um time no qual todos costumam se comunicar pouco, pode até atrapalhar o andamento das atividades.

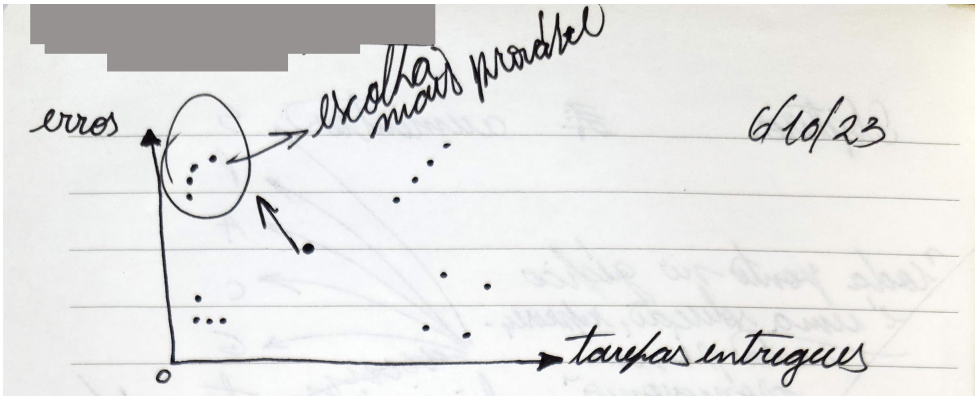
2.8.1.4 Há um tempo de integração na equipe

Há o reconhecimento de que existe um tempo mínimo para que cada novo desenvolvedor possa se tornar produtivo dentro de uma equipe. Esse tempo varia por desenvolvedor e por equipe. É o tempo de integração inicial entre a equipe e o novo funcionário, quando esse último interage com seus novos colegas e encontra seu espaço nas atividades do time. A produtividade do time tende a se estabilizar após esse período inicial de integração.

2.8.2 Sobre os times da UDS

Em 06/10/2023 uma nova conversa informal foi realizada com um dos gestores da UDS. O início dessa conversa foi marcado por um desenho à mão apresentado ao gestor (Figura 8) e uma pergunta: *para qual dos quatro times no gráfico deve ser alocado o novo desenvolvedor?*

Figura 8 – Imagem à mão apresentada ao gestor da UDS.



Fonte: elaborado pelo autor.

O desenho à mão apresentado ao gestor consiste de um gráfico com dois eixos: *tarefas entregues* e *erros*. Os pontos no gráfico representam desenvolvedores. Ao gestor foi dito que cada um dos quatro grupos de pontos mais próximos representaria um time. A situação é propositalmente montada de forma que a variabilidade de cada um dos times é próxima. Ou seja: dentro de cada time nesse gráfico os desenvolvedores são semelhantes em termos de tarefas entregues e erros gerados. Conforme experiência anterior, sabia-se de antemão que a variabilidade dentro do time nem sempre é algo desejável (Seção 2.8.1.3).

Também foi dito ao gestor que o ponto no centro do gráfico representaria um novo desenvolvedor. Da mesma forma, foi proposital o fato de que o novo desenvolvedor tem características medianas em relação aos desenvolvedores mais antigos na empresa.

Além disso, foram escolhidas apenas duas características (eixos), a fim de facilitar o entendimento do problema e o foco do gestor na questão proposta. Os eixos escolhidos representam conceitos altamente genéricos, de tarefas entregues e erros. Tarefas entregues podem ser: pequenos ajustes realizados em código, artefatos de análise, grandes projetos, etc. Erros podem ser: *bugs* em produção gerados após o trabalho do desenvolvedor, erros apontados por uma equipe de testes após avaliar tarefas, erros de interpretação ao implementar uma tarefa, etc. A semântica final desses termos ficou a cargo da interpretação do gestor. No entanto, a escolha dos eixos teve como propósito usar duas dimensões SPACE diferentes (Seção 2.1). A saber: *Performance* (na forma de erros) e *Atividade* (expressa em tarefas entregues).

Após visualizar a imagem e receber uma breve contextualização (de acordo com o descrito acima) o gestor foi rápido ao escolher: o novo desenvolvedor deveria ser alocado **no time com mais erros e menos tarefas** (o grupo de pontos mais acima, à esquerda de quem olha para o gráfico na Figura 8). Por que não escolher as outras três opções? O

gestor prefere equilibrar os times entre si, aumentando potencialmente o nível de atividade (a capacidade de entrega) do time menos produtivo, bem como aumentar sua performance (adicionando um desenvolvedor mais cuidadoso em termos de geração de erros). Na prática, durante a conversa ficou claro que o desejo do gestor é **diminuir a variabilidade entre os times** aumentando as habilidades daqueles times com menores avaliações de produtividade.

Dessa conversa com o gestor também foram obtidas outras informações relevantes sobre a formação de times na UDS:

- A quantidade máxima de vagas para desenvolvedores na UDS é fixa;
- Após o término do concurso, há chamamentos somente até que todas as vagas disponíveis sejam preenchidas;
- Um time pode receber mais do que um dos novos desenvolvedores aprovados em concurso;
- Não há limite para a quantidade máxima de desenvolvedores em um time;
- A quantidade de desenvolvedores em um time é proporcional à carga de trabalho assumida por ele.

2.8.2.1 Alocação é discutida em diferentes níveis de liderança

A partir desta entrevista informal aprendeu-se também que a escolha da equipe para a qual um novo desenvolvedor é alocado hoje passa pela Superintendência, que repassa uma lista para seus gerentes, perguntando sobre o interesse deles em relação às pessoas disponíveis para alocação. Os gerentes, por sua vez, consideram questões como:

- Há times nos quais a adição de um novo desenvolvedor não representa um desafio excessivo para o novo entrante, dada a existência de sistemas bem estabelecidos, funcionais e deveras relevantes para o Banco?
- Há alguma equipe com poucos desenvolvedores e que, devido a isso e ao nível de complexidade de seus sistemas, precise de um desenvolvedor mais experiente?

2.8.3 Acesso negado aos dados

O acesso aos dados anonimizados de produtividade de desenvolvedores foi negado em diferentes momentos e por diferentes razões. As principais razões apresentadas pelo banco para a negação de acesso foram:

- A Lei Geral de Proteção de Dados (LGPD)(BRASIL, 2018);

- O fato de que um funcionário (o autor do estudo) não pode conhecer, sob hipótese alguma, os dados de produtividade de outros funcionários;
- Não há um Termo de Consentimento Livre e Esclarecido (TCLE) assinado por cada um dos desenvolvedores envolvidos no estudo.

Tencionava-se obter dados de produtividade como, por exemplo, quantidade de *commits* por desenvolvedor, projetos nos quais o desenvolvedor trabalhou, quantidade de incidentes da equipe e quantidade de implantações da equipe. Também foram solicitados dados de cadastro dos desenvolvedores e da equipe, como tempo no qual o desenvolvedor está na equipe, data de nascimento, data de admissão e cargo. Sobre os novos desenvolvedores, solicitou-se a nota da prova final de cada um no período de treinamento, bem como o código fonte e os testes automatizados criados por eles nessa prova.

Inicialmente, procurava-se obter a quantidade máxima de dados possível para que a análise exploratória de dados pudesse fornecer um ponto de partida sobre qual abordagem utilizar no problema de alocação, focando-se no aumento da produtividade após a indicação do novo desenvolvedor para a equipe. No entanto, dado o cuidado da empresa com os dados de produtividade dos desenvolvedores e a consequente negativa de fornecimento deles para o estudo, passou-se a buscar uma **fonte de dados** alternativa para a viabilização do trabalho.

2.9 Fonte de dados

Com o objetivo de viabilizar o estudo, buscou-se uma fonte de dados que fosse aberta e, ao mesmo tempo, associada a alguma empresa tradicional no campo de produção de software. Escolheu-se então o grupo de projetos de código aberto associados à plataforma *.NET* (leia-se "*dot net*"). Tais projetos estão disponíveis publicamente no endereço github.com/dotnet e são geridos pela empresa *Microsoft* (microsoft.com).

A disponibilização pública dos dados da plataforma .NET é realizada no *GitHub* (github.com). Um serviço bastante conhecido entre desenvolvedores de software; também muito utilizado por projetos de código aberto. No GitHub, além de empresas, pessoas físicas comuns também podem criar seus próprios repositórios.

O GitHub oferece a seus usuários uma *API REST* (docs.github.com/en/rest) a partir da qual diversos tipos de dados podem ser obtidos de forma simplificada. Podem ser extraídas listas de repositórios associados a uma determinada organização, bem como dados dos *contributors*, *issues* e *commits* de cada um desses repositórios, além de muitos outros dados. A título de exemplo, caso o interesse seja acessar os dados da organização .NET e do projeto *ASP.NET Core* sob sua responsabilidade, os seguintes endereços podem ser utilizados:

- **Repositórios .NET**
  - **Endpoint:**
    - api.github.com/orgs/**dotnet**/repos
  - **Documentação:**
    - docs.github.com/en/rest/repos/repos#list-organization-repositories
- ***Contributors ASP.NET Core***
  - **Endpoint:**
    - api.github.com/repos/**dotnet**/**aspnetcore**/contributors
  - **Documentação:**
    - docs.github.com/en/rest/repos/repos#list-repository-contributors
- ***Issues ASP.NET Core***
  - **Endpoint:**
    - api.github.com/repos/**dotnet**/**aspnetcore**/issues
  - **Documentação:**
    - docs.github.com/en/rest/issues/issues#list-repository-issues
- ***Commits ASP.NET Core***
  - **Endpoint:**
    - api.github.com/repos/**dotnet**/**aspnetcore**/commits
  - **Documentação:**
    - docs.github.com/en/rest/commits/commits#list-commits

3 MATERIAIS E MÉTODOS

> O crescimento do conhecimento marcha de velhos problemas para novos por intermédio de conjecturas e refutações.(MARCONI; LAKATOS, 2021, p. 96)

Este capítulo do trabalho descreve o uso do aprendizado adquirido no Capítulo 2. O início resume-se na proposta de um método adequado ao estudo. Posteriormente, descreve-se até o fim do capítulo a aplicação prática do método proposto.

3.1 Método

Propõe-se aqui uma pesquisa com paradigma *positivista*(SORDI, 2013, p. 98): *quantitativa*, *objetiva* e *experimental*. Realizada com base no método *hipotético-dedutivo*, com o qual busca-se a verdade partindo-se de um *problema* cuja tentativa de resolução é dada por uma solução provisória (*conjectura*). Esta última denominada *teoria-tentativa*, a qual é colocada à prova através de críticas tendo como objetivo a *eliminação do erro* (MARCONI; LAKATOS, 2021, p. 94). Dessa forma, no contexto deste estudo:

- **Problema** é o processo de escolha do time para o qual será alocado um novo desenvolvedor, que nada mais é do que um POM (Seção 2.3);
- **Conjectura** (ou *teoria-tentativa*) é cada um dos conjuntos iniciais de hiperparâmetros associados ao NSGA-II (Seção 2.5);
- **Eliminação do erro** é a busca por valores cada vez maiores de hipervolumes (Seção 2.6) associados às fronteiras de Pareto (Seção 2.2) encontradas para o POM.

Com relação ao POM definido acima, de forma mais detalhada, deseja-se **minimizar a variabilidade da produtividade** entre os times de desenvolvimento (Seção 2.8.2), dado que a produtividade de um time pode ser expressa pelas dimensões SPACE (Seção 2.1). Além disso, os valores atribuídos às dimensões SPACE de um time de desenvolvimento são o resultado da soma do valor estimado de cada uma delas para cada desenvolvedor que faz parte do time.

Definido o método a ser utilizado, realizou-se a **preparação dos dados** para utilização no estudo.

3.2 Preparação dos dados

A preparação dos dados foi realizada a fim de se criar um conjunto de dados para aplicação do método proposto na Seção 3.1. Extraiu-se da fonte de dados apresentada na Seção 2.9 os dados brutos que seriam utilizados no estudo (Seção 3.2.1), compilou-se esses dados em arquivos intermediários (Seção 3.2.2) e, por fim, um conjunto de dados final foi montado (Seção 3.2.3).

3.2.1 Extração de dados brutos

Foram realizadas sessões de extração de dados a fim de se obter dados dos desenvolvedores associados aos projetos .NET. As sessões de extração consistiram de acessos às APIs do GitHub (Seção 2.9). As chamadas aos *endpoints* do GitHub foram convertidas em arquivos JSON. Cada um dos arquivos JSON criados é resultado de um único acesso, a um único *endpoint*, e representa apenas uma parte (*página*[1]) do conjunto de resultados desejado.

O primeiro conjunto de dados obtido consistiu de (1) 9 arquivos JSON com dados de todos os repositórios da organização .NET (30 por página). Os nomes de repositórios, obtidos a partir destes primeiros arquivos, foram utilizados para obtenção de (2) 19.189 arquivos JSON com dados de *commits* (100 por página), (3) 346 arquivos JSON com dados de *contributors* (100 por página) e (4) 8.026 arquivos JSON com dados de *issues*.

3.2.2 Compilando dados

O conteúdo de cada arquivo JSON obtido a partir dos *endpoints* da API do GitHub foi utilizado para a geração de arquivos auxiliares, em formato CSV. Foram gerados arquivos CSV específicos para: (1) os nomes de todos os 246 repositórios cujo proprietário é a organização .NET (*dotnet*), (2) os nomes de 17 *bots* identificados durante o processo de compilação dos dados dos arquivos JSON originais, (3) os dados de 1.110.429 commits realizados nos repositórios .NET, (4) dados de 629.442 *issues*, também associadas aos repositórios citados.

Esses arquivos CSV são uma primeira compilação e filtragem dos dados contidos nos arquivos JSON originais e serviram de base para a elaboração de um **conjunto de dados intermediário**, com as **estimativas normalizadas de 3 dimensões SPACE** calculadas para desenvolvedores associados aos repositórios em estudo. O significado das colunas referentes a essas estimativas de dimensões SPACE estão listadas abaixo.

[1] É comum que APIs REST ofereçam a funcionalidade de paginação quando há uma quantidade grande de resultados a serem retornados. Também é comum que se defina um tamanho máximo para a *página*. No caso da API do GitHub esse limite é de 100 registros na maioria dos *endpoints* consumidos.

- **SATISFACAO**: quantidade de dias entre as datas do primeiro e do último commit;
- **ATIVIDADE**: média da quantidade de *commits* por mês;
- **COMUNICACAO**: quantidade de *issues* com as quais o desenvolvedor está relacionado.

O conjunto de dados intermediário é composto por 10.554 linhas e contém, além das 3 dimensões SPACE citadas acima, um **identificador** para o desenvolvedor (gerado arbitrariamente pelo autor), o ***login*** utilizado pelo desenvolvedor no GitHub e o **repositório** ao qual o desenvolvedor está associado. As primeiras linhas desse conjunto de dados intermediário podem ser observadas na Tabela 5 (a coluna com o *login* do desenvolvedor foi eliminada para preservar a identidade dos perfis analisados).

Tabela 5 – Primeiras linhas do conjunto de dados intermediário que, no total, possui 10.554 linhas.

ID	REPOSITÓRIO	SATISFACAO	ATIVIDADE	COMUNICACAO
Dev_0	roslyn	0,268889	1,000000	0,666392
Dev_1	sdk	0,084727	0,507254	0,023476
Dev_2	dotNext	0,304136	0,476023	0,007963
Dev_3	sdk	0,053671	0,430414	0,055327
Dev_4	vscode-dotnet-runtime	0,070798	0,388352	0,024986
(...)				

Fonte: elaborado pelo autor.

Cada linha apresentada na Tabela 5 refere-se a um único desenvolvedor. Como há desenvolvedores com interações em mais de um repositório, então adotou-se um critério simples para o preenchimento dessa coluna: o repositório escolhido é aquele no qual o desenvolvedor fez seu último *commit*.

Na Tabela 5 também pode-se observar, por exemplo, que o *Dev_0* é um desenvolvedor com a estimativa máxima de ATIVIDADE (com valor "1,000000"). Como as estimativas foram normalizadas com o uso de *MinMaxScaler*[2], da biblioteca *Scikit-Learn*[3], o valor máximo de cada uma das colunas de estimativa é 1 (um), valor que é atribuído ao nível de ATIVIDADE do *Dev_0*. Vale destacar aqui que o nível de atividade 0 (zero) não significa que um desenvolvedor é inativo. Significa apenas que ele é o desenvolvedor com o menor nível de atividade, que não necessariamente é nula.

[2] Documentação em scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
[3] scikit-learn.org

3.2.3 Conjunto de dados final

Para criar o **conjunto de dados final**, aquele que foi efetivamente utilizado no estudo, optou-se pelo destaque de dados reais de desenvolvedores que realmente estivessem comprometidos com os repositórios com os quais estão envolvidos. Dessa forma, utilizou-se o conjunto de dados intermediário (obtido conforme descrito na Seção 3.2.2) como base, filtrando-se os desenvolvedores com o uso de um critério de relevância em relação à média dos valores estimados para cada uma das dimensões SPACE avaliadas.

Foram selecionados apenas registros de **desenvolvedores cujos 3 valores estimados estivessem acima da soma da média com o desvio padrão** da coluna referente a cada uma das estimativas SPACE. Essa etapa de filtragem diminuiu o tamanho da listagem de estimativas de desenvolvedores de 10.554 para 180 registros.

O conjunto de dados final, assim como o conjunto intermediário, referencia o último repositório no qual o desenvolvedor realizou *commits*. Devido a isso, também foi possível eliminar as referências a repositórios em alguns dos registros do conjunto de dados final, a fim de criar uma situação problema na qual pudesse ser realizada a escolha de qual o melhor repositório para alocação de um novo desenvolvedor.

No conjunto de dados final, os 180 desenvolvedores estavam distribuídos em um total de 60 repositórios diferentes, sendo que 33 repositórios estavam associados a somente um único desenvolvedor. *Por que não assumir que esses* 33 *repositórios serão descontinuados e planejar, hipoteticamente, a realocação de tais desenvolvedores nos outros* 27 *repositórios restantes?* Essa foi a escolha realizada neste trabalho. Resolver o problema hipotético de realocar 33 desenvolvedores em 27 repositórios compostos de, pelo menos, 2 desenvolvedores cada.

A partir deste ponto extrapola-se o conceito de *repositório* no conjunto de dados final, doravante chamado de *time*. Assume-se que os desenvolvedores do conjunto de dados final formam times, o que não é necessariamente verdade. É evidente, no entanto, que existe um grau de conexão entre desenvolvedores que participam de um mesmo projeto de código aberto, mesmo que essa conexão não esteja formalizada em um contrato de trabalho.

Por fim, caso exista interesse do leitor, o CSV com o conjunto de dados final pode ser obtido diretamente do repositório criado em **github.com/brucelucien/2023-notario-mba-tcc-dados-publicos** exclusivamente para esse fim. Nesse diretório, o arquivo com os dados do conjunto de dados final é aquele denominado ***space_dataset_final_anonimizado.csv***.

3.3 Modelagem

Aqui o estudo apoia-se diretamente nos resultados da revisão bibliográfica (Seção 2.4.6). Dado que a questão de pesquisa (Seção 1.1) é direcionada à criação de um modelo que indique o *time ideal* para o qual um novo desenvolvedor deve ser alocado, será utilizado o algoritmo mais popular entre os trabalhos acadêmicos encontrados na revisão bibliográfica: NSGA-II (DEB *et al.*, 2002).

3.3.1 Definição da abordagem

Neste estudo assume-se que **cada solução é um indivíduo** de uma **população de soluções** que evolui **de geração a geração** (ou em *iterações*). **As melhores soluções são aquelas com mais aptidão** (avaliadas com o uso de uma **função de aptidão**), selecionadas por meio de um **mecanismo específico de seleção**. Uma determinada solução, ou **indivíduo**, pode ser gerada de forma puramente aleatória (na *população inicial*) ou a partir de outras duas soluções, com o uso de **crossover** (*recombinação*); além disso, uma solução também pode ser alterada por meio de **mutação**.

3.3.2 Formato de uma solução

Inspirado na estrutura de *cromossomo* apresentada em Garshasbi *et al.* (2019), este estudo parte da suposição de que, em uma solução hipotética para o problema proposto (ilustrada na Tabela 6), os desenvolvedores disponíveis para alocação são posições fixas de um vetor. A esses desenvolvedores são atribuídos os nomes dos times para os quais serão alocados. Portanto, cada indivíduo dentro das populações geradas pelo NSGA-II terá essa estrutura, adequada para o uso da biblioteca pymoo (Seção 2.7), utilizada na próxima seção.

Tabela 6 – Solução hipotética.

#	DESENVOLVEDOR	TIME
0	Desenvolvedor 1	Time L
1	Desenvolvedor 2	Time H
(...)	(...)	(...)
$N-1$	Desenvolvedor N	TIME W

Fonte: elaborado pelo autor.

O cálculo da função de aptidão também é baseado nessa estrutura e simplesmente simula a presença dos desenvolvedores nos times que já existem na empresa. Calcula-se o *desvio padrão*, **entre os times**, da **soma do valor estimado de todos os desenvolvedores em cada time** (para SATISFACAO, ATIVIDADE e COMUNICACAO). Sendo assim, a função de aptidão aplicada sobre uma solução qualquer retornará 3 valores de *desvio padrão*, um para cada característica estimada.

3.3.3 Implementação

Utilizou-se a biblioteca pymoo (Seção 2.7). Resumidamente, a implementação consistiu de: (1) estender a classe *ElementwiseProblem*, (2) implementar nela o *construtor* e o método *_evaluate*, (3) instanciar a classe *NSGA2*, (4) definir um critério de término e, por fim, (5) executar a função *minimize* (que realiza a minimização propriamente dita).

O *construtor* foi implementado para receber um *DataFrame* composto pela totalidade dos desenvolvedores da empresa. Tanto aqueles já participantes de um time de desenvolvimento quanto aqueles com situação indefinida. É nesse método que há uma extração de 3 conjuntos de dados a partir do *DataFrame* passado como parâmetro: uma lista de times de desenvolvimento e dois novos *DataFrames*, um com os desenvolvedores atuais, já integrantes de um time, e outro com os desenvolvedores novos, sem time atribuído.

É no *construtor* também que define-se tanto a quantidade de variáveis do problema (cada solução é um conjunto de números aleatórios cuja cardinalidade é esta definida aqui) e a quantidade de objetivos. No contexto deste trabalho, o número de variáveis é a quantidade de times de desenvolvimento. O número de objetivos é 3, um para cada uma das dimensões SPACE disponíveis no conjunto de dados final (Seção 3.2.3). A saber: SATISFACAO, ATIVIDADE e PERFORMANCE.

A implementação do método *_evaluate*, que é a função de aptidão propriamente dita, consistiu do cálculo de variabilidade de cada uma das dimensões SPACE citadas no final do parágrafo anterior. Este método recebe como parâmetro um conjunto de números inteiros, de 0 (zero) a $N - 1$, onde N é a quantidade de times, que foi definida no *construtor*. Conforme descrito na Seção 3.3.2, assume-se então uma listagem na qual há posições fixas de desenvolvedores sem time associado e os times são atribuídos a esses desenvolvedores seguindo a posição sugerida pelo conjunto de números passado como parâmetro. O que se faz então é calcular o *desvio padrão* desta solução completa, considerando os desenvolvedores antigos e os novos, estes últimos colocados nos times sugeridos pela sequência de números passada como parâmetro.

Estendida a classe *ElementwiseProblem*, passa-se à instanciação do algoritmo que será utilizado. Neste trabalho instanciou-se a classe *NSGA2*, referente ao algoritmo NSGA-II. É nessa instanciação que define-se (1) o tamanho da população inicial, (2) a quantidade de descendentes por geração, (3) a forma de amostragem das soluções (utilizou-se a classe *PermutationRandomSampling*), (3) o método de *crossover* (escolheu-se *OrderCrossover*), (4) o método de mutação (instanciou-se *InversionMutation*) e, por fim, (4) se há eliminação de soluções duplicadas ou não (neste trabalho optou-se por sim; os descendentes duplicados são eliminados).

As instanciações de parâmetros citadas no parágrafo anterior referem-se a *operadores genéticos*. Foram escolhidas de tal forma que mantém a estrutura das soluções intacta,

conforme comentado anteriormente sobre permutação (Seção 2.7). Além disso, as escolhas são semelhantes àquelas mostradas por Garshasbi *et al.* (2019), um trabalho que apresenta uma estrutura de *cromossomo* que também se adéqua ao problema em estudo aqui.

O critério de término é realizado com o chamamento da função *get_termination* e foi definido como o limite da quantidade de gerações geradas pelo algoritmo.

As configurações anteriores são exigência para a utilização da função *minimize*. Essa função recebe como parâmetro (1) o problema e (2) o algoritmo instanciados, (3) o critério de término e alguns parâmetros opcionais. Nesse trabalho foram utilizados os parâmetros opcionais $seed = 42$ (maior controle dos resultados), $save\_history = True$ (permite análise posterior dos resultados por geração) e $verbose = True$ (registro da evolução de cada geração).

Resta ainda comentar que, com o intuito de facilitar a comparação entre diferentes conjuntos de soluções, neste estudo o valor do hipervolume foi normalizado, podendo variar dentro do intervalo $[0, 1]$. A normalização foi realizada automaticamente com o uso do parâmetro *zero_to_one=True* na instanciação da classe *Hypervolume*.

A implementação descrita acima, nesta seção, pode ser visualizada integralmente no arquivo ***experimento.ipynb***, disponível no repositório **github.com/brucelucien/2023-notario-mba-tcc-dados-publicos**.

3.3.4 Execução dos experimentos

A busca pelo melhor conjunto de soluções possíveis para o problema proposto foi realizada na forma de 37 experimentos, todos realizados com parâmetro aleatório fixo ($seed = 42$), conforme apresentado na Seção 3.3.3. Os experimentos serão detalhados na Seção 4.5.1.

Inicialmente, a ordem na qual os experimentos foram realizados foi arbitrária. O valor do hipervolume retornado por uma execução guiou alterações posteriores nos hiperparâmetros. O objetivo foi o de usar *exploitation* para obter valores próximos dos ideais para os parâmetros e *exploration* para verificar os efeitos de alterações nas fronteiras de tais valores.

Uma das primeiras configurações testadas considerava uma população de 100 indivíduos, taxa de descendentes $0,7$ e 100 gerações. Esta configuração, uma das 10 melhores entre todas as tentativas, permaneceu relevante até o final do estudo. Buscou-se melhorá-la com a variação dos valores dos parâmetros para mais ou para menos. Como o valor do hipervolume das primeiras tentativas não foi superado, buscou-se avaliar outras configurações, com populações maiores e maior número de gerações. Dessa forma chegou-se nas configurações com 1.000 e 1.500 na população inicial, que também estão entre as 10 melhores configurações encontradas. Por fim, configurações com 500 indivíduos na população inicial apresentaram resultado superior em relação às demais configurações

testadas.

4 RESULTADOS E DISCUSSÕES

4.1 Lições aprendidas com o negócio

O esforço por um melhor entendimento sobre a realidade organizacional da UDS e a busca pelos dados que seriam utilizados no estudo (Seção 2.8) levaram às seguintes conclusões:

1. A experiência profissional anterior dos novos desenvolvedores pode ser um bom indicador de conhecimento sobre determinados domínios de atuação, mas talvez não seja um bom indicador de conhecimento e/ou competência na aplicação das tecnologias utilizadas pelo banco (Seção 2.8.1.1).

2. Os três meses iniciais de treinamento representam uma oportunidade valiosa para análise do perfil dos novos desenvolvedores mas, em termos de tecnologia específica, só podem avaliar a atuação deles em relação ao *framework* interno do Banrisul (Seção 2.8.1.2).

3. Supor que existe uma necessidade de agregar ao time características que ele não tem nem sempre vai de acordo com o desejado pelos líderes, o que exige uma abordagem flexível quanto à estratégia de otimização escolhida para a escolha do time no qual o novo desenvolvedor será alocado (Seção 2.8.1.3).

4. Avaliações sobre o impacto de um novo desenvolvedor em um time pré-existente precisam considerar que existe um período de integração entre o novo funcionário e o time. A real contribuição deste novo desenvolvedor para a produtividade do time deve ser uma comparação entre o momento anterior ao ingresso no time e o momento no qual a produtividade do time volta a se estabilizar (Seção 2.8.1.4).

5. Estratégias de avaliação da produtividade do time após a alocação de um novo desenvolvedor precisam oferecer certo nível de facilidade de alteração e flexibilidade para que avaliações qualitativas dos líderes possam ser consideradas (Seção 2.8.2.1).

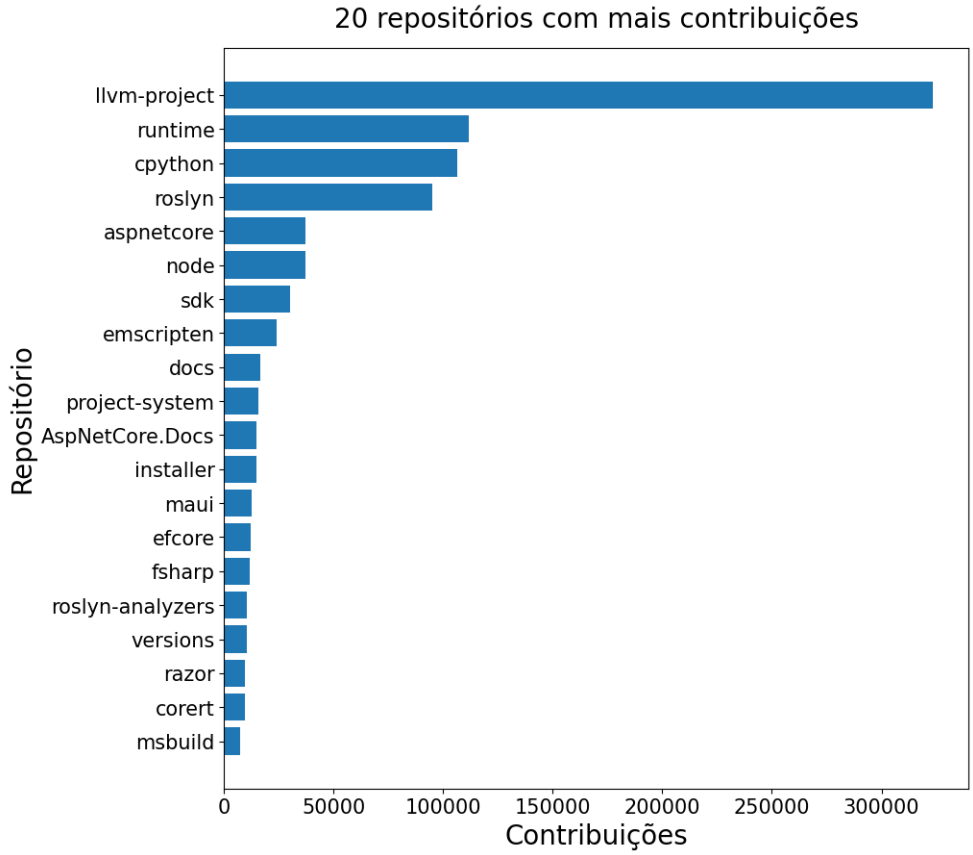
4.2 Fatos sobre os dados utilizados

A análise exploratória, realizada em diferentes momentos durante o entendimento e a preparação dos dados, revelou alguns fatos dignos de citação dada sua relevância para o entendimento do contexto no qual o GitHub está inserido. Tais fatos são apresentados nesta seção.

4.2.1 Os 20 projetos com mais contribuições

Os dados de *contributors* de um projeto contém um campo chamado *contributions*, que é preenchido com números inteiros. A Figura 9 foi gerada com o uso dos dados desse campo. O gráfico na figura mostra, de forma ordenada, quais são os 20 repositórios .NET com mais contribuições. O objetivo aqui é mostrar graficamente o grau de interesse da comunidade de código aberto .NET nos projetos em questão.

Figura 9 – 20 repositórios com mais contribuições.



Fonte: elaborado pelo autor.

A forma do gráfico parece sugerir que o interesse nos repositórios .NET obedece a uma *Lei de Potência*. Pode-se notar também que o repositório *llvm-project* (github.com/dotnet/llvm-project) possui um número de contribuições muito superior ao do segundo repositório, o repositório *runtime* (github.com/dotnet/runtime).

Destaca-se o fato de que, ao acessar a página do repositório *llvm-project*, pode-se notar que ele é, na realidade, um *fork* de um repositório homônimo do *owner llvm* (acessível em github.com/llvm/llvm-project). O repositório *runtime*, pelo contrário, é do próprio *owner* .NET, cujos dados de repositórios estão sendo avaliados neste estudo. Esse detalhe do *fork* pode impactar a interpretação dos resultados do trabalho atual. Contudo, como

o foco do estudo não é a avaliação da API do GitHub, não será considerado. A questão revela detalhes de implementação bastante específicos da API do GitHub, que pode estar retornando para o *fork* os dados do projeto original.

4.2.2 Relação entre contribuição e quantidade de desenvolvedores

Foram analisadas em conjunto, e por repositório, a **quantidade de contribuições** e a **quantidade de desenvolvedores**. Algumas medidas descritivas dessa análise podem ser visualizadas na Tabela 7. Destaca-se nessa tabela o fato de que os indícios de variabilidade (o desvio padrão, no caso) das contribuições por repositório são muito superiores em relação à média (*Coeficiente de Variação* $\approx 535,37$). Os indícios de variabilidade dos desenvolvedores também são altos, mas menos impactantes (*Coeficiente de Variação* $\approx 156,52$). Outro dado que se destaca é a amplitude entre o valor mínimo e máximo de contribuições por repositório. Enquanto há repositórios com apenas uma contribuição, o repositório com mais contribuições recebeu mais de 300.000.

Tabela 7 – Medidas descritivas das quantidades, de contribuições e de desenvolvedores, por repositório.

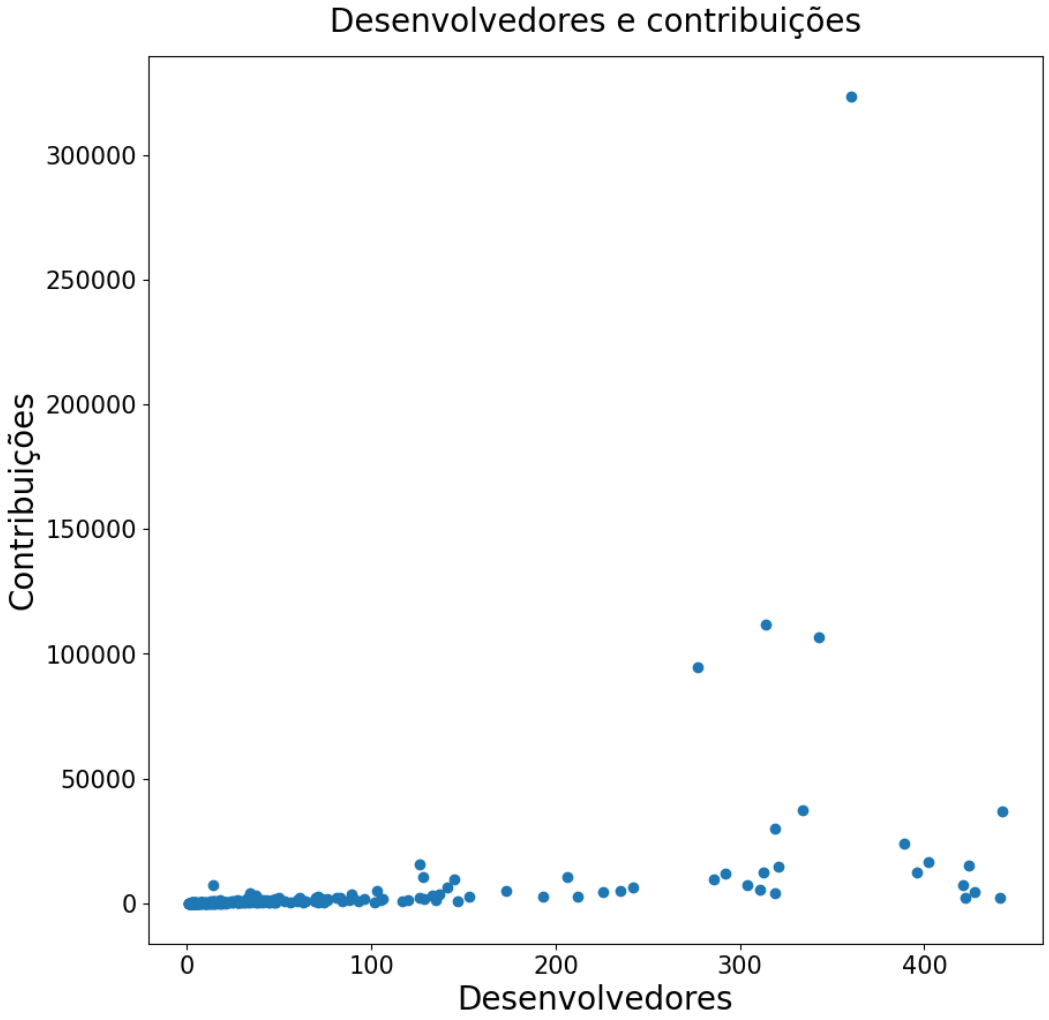
Medida	**Contribuições**	**Desenvolvedores**
contagem	247,00	247,00
média	4.444,78	65,93
desvio padrão	23.796,10	103,19
min	1,00	1,00
25%	85,50	8,00
50%	320,00	20,00
75%	1.422,50	71,00
max	323.611,00	442,00

Fonte: elaborado pelo autor.

Intuitivamente pode-se supor que a quantidade de contribuições por repositório tem relação direta (linear e crescente) com a quantidade de desenvolvedores mobilizada pelo repositório. Não é exatamente isso que os dados revelam. Calculando-se o *coeficiente de Pearson* dessa correlação chega-se ao valor de $0,423656$. Uma correlação, no máximo, moderada. No gráfico da Figura 10 pode-se notar que há repositórios cuja quantidade de contribuições é muito superior à média dos demais repositórios.

Obtém-se os *outliers* listados na Tabela 8 quando assume-se como *outliers* os repositórios cujas quantidades de contribuições estão acima do valor da média acrescida de um desvio padrão. Os quatro primeiros repositórios listados nessa tabela são os quatro pontos mais destacados na Figura 10. Os outros não parecem relevantes na análise a partir do gráfico, mas correspondem à regra de filtragem utilizada. A percepção de destaque deles

Figura 10 – Relação entre desenvolvedores e contribuições em cada repositório.



Fonte: elaborado pelo autor.

em relação aos outros projetos é prejudicada justamente pelos *outliers* mais significativos, como é o caso das quantidades de contribuições do repositório *llvm-project*.

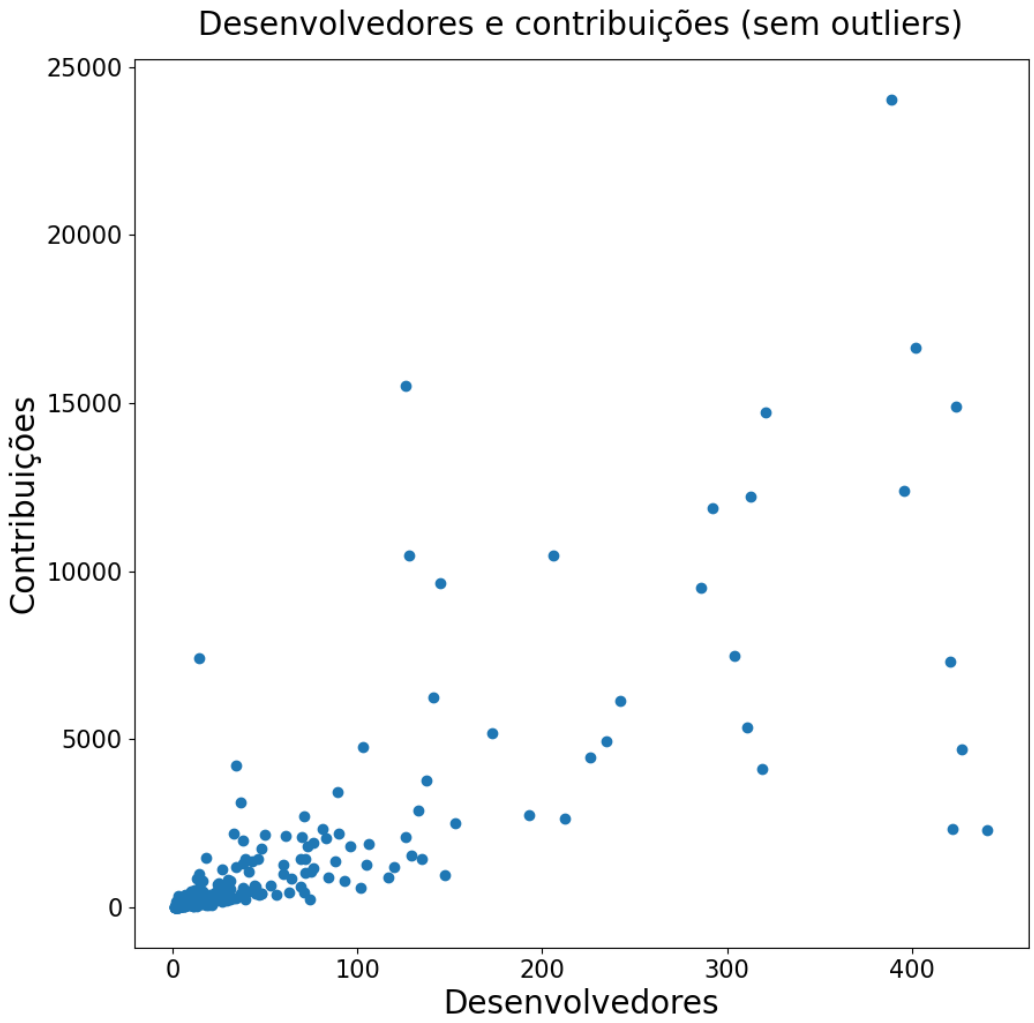
Tabela 8 – Projetos considerados *outliers* em relação à quantidade de contribuições.

Repositório	**Contribuições**	**Desenvolvedores**
llvm-project	**323.611**	**360**
runtime	**111.636**	**314**
cpython	**106.505**	**343**
roslyn	**94.820**	**277**
aspnetcore	37.347	334
node	36.943	442
sdk	30.077	319

Fonte: elaborado pelo autor.

Quando os *outliers* listados na Tabela 8 são retirados do conjunto de dados analisado, obtém-se o gráfico da Figura 11. O *coeficiente de Pearson* entre as variáveis aqui é de $0,770959$, uma correlação que já pode ser considerada forte. Nesse gráfico, confirma-se a intuição de que há uma relação linear entre quantidade de desenvolvedores e volume de contribuições. No entanto, nota-se também que há no gráfico um aumento da variabilidade no número de contribuições conforme cresce a quantidade de desenvolvedores envolvidos com o repositório.

Figura 11 – Relação entre desenvolvedores e contribuições em cada repositório após a retirada de outliers.



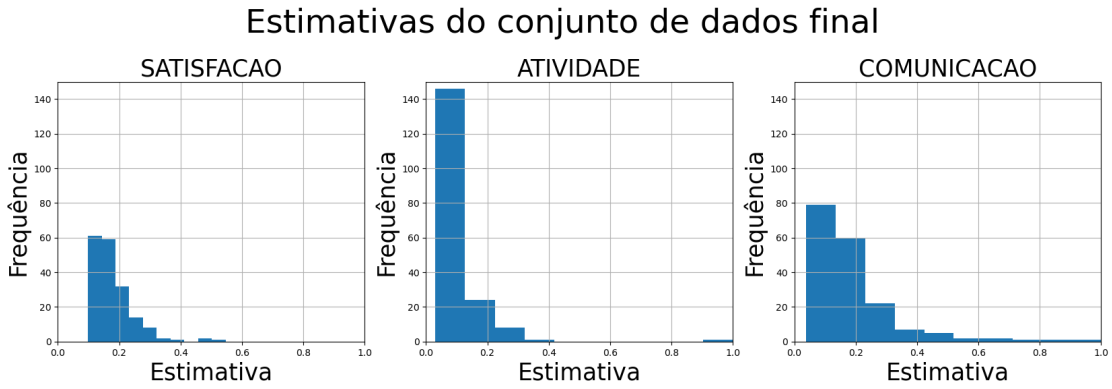
Fonte: elaborado pelo autor.

4.3 Características do conjunto de dados final

A Figura 12 mostra os histogramas associados a cada uma das estimativas avaliadas dos 180 desenvolvedores listados no conjunto de dados final. A partir desses histogramas

pode-se inferir que os desenvolvedores se comunicam e colaboram entre si através de *issues* (COMUNICACAO) em uma escala maior do que seu nível de entrega de *commits* (ATIVIDADE). Também segundo esses histogramas, os desenvolvedores selecionados parecem ter um nível de envolvimento com os projetos (SATISFACAO) semelhante a seu nível de envolvimento em *issues* (COMUNICACAO), o que implica que seu tempo dedicado aos projetos (SATISFACAO) é também maior do que seu nível de realização de *commits* (ATIVIDADE).

Figura 12 – Estimativas do conjunto de dados final.



Fonte: elaborado pelo autor.

A Tabela 9 mostra algumas estatísticas descritivas do conjunto de dados final. A comparação entre as médias de SATISFACAO, ATIVIDADE e COMUNICACAO confirma os comentários anteriores baseados nos histogramas. O desvio padrão maior da estimativa de COMUNICACAO em relação às outras estimativas pode indicar que os desenvolvedores são bem mais diversos em termos de envolvimento nas discussões coletivas do que em seus padrões de envolvimento (SATISFACAO) e realização de *commits* (ATIVIDADE).

Tabela 9 – Medidas descritivas do conjunto de dados final utilizado no estudo.

Medida	**SATISFACAO**	**ATIVIDADE**	**COMUNICACAO**
contagem	180,00	180,00	180,00
média	0,18	0,09	0,16
desvio padrão	0,07	0,09	0,15
min	0,10	0,03	0,04
25%	0,13	0,04	0,09
50%	0,16	0,06	0,15
75%	0,20	0,11	0,22
max	0,55	1,00	1,00

Fonte: elaborado pelo autor.

Talvez estar envolvido no projeto e realizar alterações de código sejam requisitos básicos para desenvolvedores relevantes em seus repositórios; envolvimento em discussões, nem tanto. Resta ainda um comentário sobre o máximo de SATISFACAO mostrado na Tabela 9, que é aproximadamente a metade do nível estimado de ATIVIDADE e de COMUNICACAO. Essa característica do conjunto de dados final pode indicar que é mais importante para desenvolvedores atuantes estar envolvido em discussões e alterações de código do que investir na participação, no mesmo projeto, durante muito tempo.

Neste ponto troca-se a nomenclatura de *repositório* para *time*, repetindo-se a extrapolação de conceito comentada e realizada na Seção 3.2.3.

As quantidades finais de desenvolvedores por time estão listadas na Tabela 10. Nessa tabela, o time apresentado como *INDEFINIDO* é utilizado para apresentar a quantidade de desenvolvedores que anteriormente eram os únicos desenvolvedores relevantes em um repositório. Esses são os 33 desenvolvedores cujas estimativas serão utilizadas para avaliação de qual seria o melhor time para o qual alocá-los.

Tabela 10 – Quantidade de desenvolvedores por time no conjunto de dados final.

Time	**Quantidade de desenvolvedores**
INDEFINIDO	33
runtime	25
roslyn	15
maintenance-packages	11
aspnetcore	9
docs	9
aspire	7
maui	6
AspNetCore.Docs	5
efcore	5
installer	5
winforms	5
arcade	5
sdk	4
fsharp	3
dotnet-api-docs	3
interactive	3
csharplang	3
msbuild	3
project-system	3
razor	3
core	3
eShop	2
EntityFramework.Docs	2
docfx	2
orleans	2
tye	2
install-scripts	2

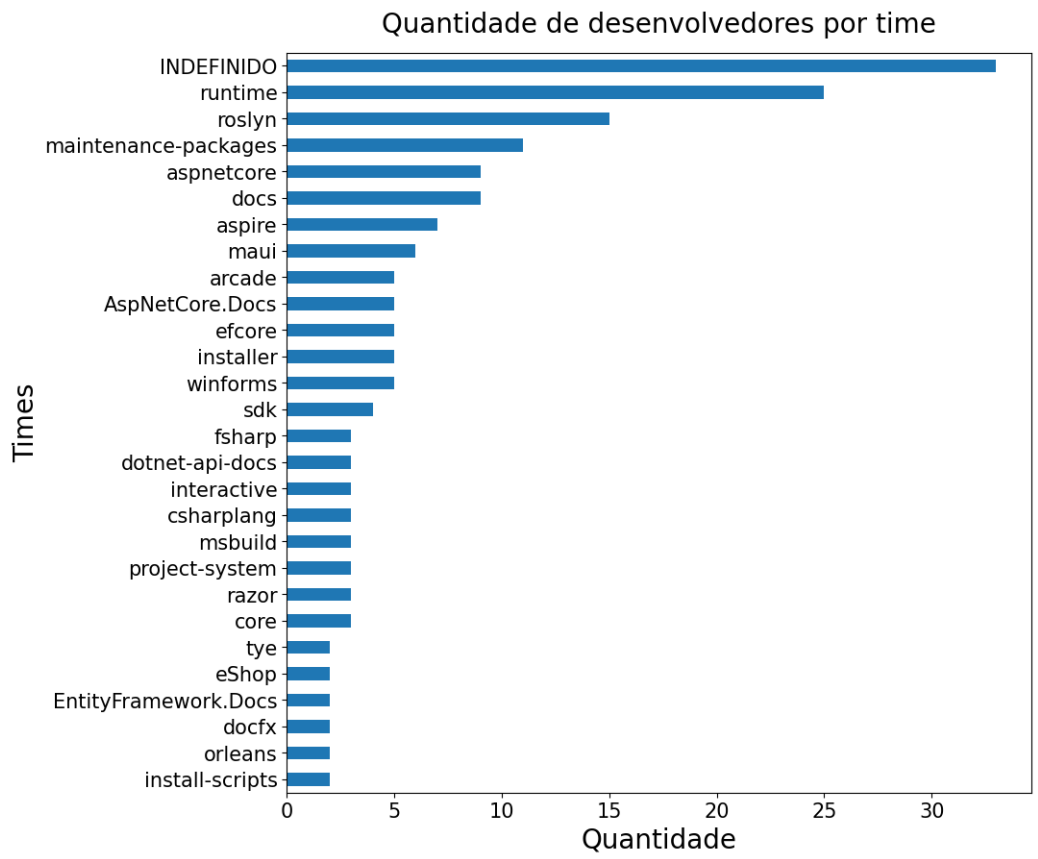
Fonte: elaborado pelo autor.

Os dados da Tabela 10 também podem ser visualizados na Figura 13, que mostra de forma gráfica a proporção de desenvolvedores em cada time, bem como a proporção de desenvolvedores com time indefinido em relação aos demais.

4.4 Solução base

A Tabela 11 foi obtida calculando-se a variabilidade (*desvio padrão*) entre os times presentes do conjunto de dados final (Seção 4.3), que será utilizado para realização de experimentos. É o valor da variabilidade sem a alocação dos desenvolvedores com situação indefinida. Essa é a pior solução admissível para o problema. Neste estudo adotou-se o nome de *solução base* para esta configuração inicial de times.

Figura 13 – Quantidade de desenvolvedores por time no conjunto de dados final.



Fonte: elaborado pelo autor.

Tabela 11 – Solução base. O estado atual da empresa, indício de variabilidade de cada dimensão, sem a inclusão de novos desenvolvedores.

SATISFACAO	ATIVIDADE	COMUNICACAO
1,375253	0,702518	1,227839

Fonte: elaborado pelo autor.

4.5 Avaliação do modelo

4.5.1 Resultados dos experimentos

Os experimentos realizados estão listados na Tabela 12, que está ordenada de forma decrescente em relação ao hipervolume para que os experimentos mais bem sucedidos sejam exibidos nas primeiras posições.

Tabela 12 – Experimentos realizados.

#	**População** *pop_size*	**Descendentes** *n_offsprings*	**Gerações** *n_gen*	**Soluções**	**Uso (%)**	**Hipervolume** *hypervolume*
1	500	150	8000	500	100,00	84,43
2	500	350	4000	500	100,00	84,32
3	500	150	4000	500	100,00	83,84
4	500	350	3000	500	100,00	83,72
5	500	350	2000	490	98,00	83,69
6	1000	700	2000	1000	100,00	82,96
7	1250	875	1000	964	77,12	82,14
8	100	70	100	55	55,00	80,86
9	1375	962	100	196	14,25	80,60
10	1000	700	500	485	48,50	80,27
11	100	30	150	30	30,00	79,96
12	1500	1050	50	101	6,73	79,33
13	1000	300	150	104	10,40	79,00
14	100	70	150	76	76,00	78,72
15	1000	700	150	168	16,80	78,37
16	1000	700	200	219	21,90	78,24
17	100	70	200	100	100,00	77,89
18	1250	875	100	183	14,64	77,57
19	680	476	230	274	40,29	77,24
20	1000	700	100	151	15,10	76,98
21	1000	900	150	244	24,40	76,96
22	100	30	10000	100	100,00	76,94
23	500	350	100	130	26,00	75,40
24	1125	788	100	149	13,24	74,97
25	100	80	150	100	100,00	74,69
26	100	70	4000	100	100,00	74,52
27	1500	1050	100	191	12,73	74,31
28	100	70	250	100	100,00	74,15
29	100	70	50	51	51,00	74,06
30	1000	700	50	82	8,20	72,68
31	500	350	50	82	16,40	72,43
32	100	90	10000	100	100,00	72,37
33	100	70	500	100	100,00	72,21
34	100	70	300	100	100,00	72,04
35	100	70	1000	100	100,00	71,60
36	100	70	2000	100	100,00	71,10
37	100	70	10000	100	100,00	70,39

Fonte: elaborado pelo autor.

Cada registro da Tabela 12 contém o **tamanho da população usada**, a **quantidade de descendentes gerados**, por **quantas gerações o algoritmo foi rodado**, a **quantidade de soluções encontradas**, o **percentual de soluções em relação ao tamanho da população** (*Uso (%)*) e o **valor normalizado do hipervolume** multiplicado por 100 (para facilitar a interpretação do indicador).

Apesar de nem sempre se confirmarem, se faz necessário comentar sobre dois tipos de padrões percebidos durante a realização dos experimentos:

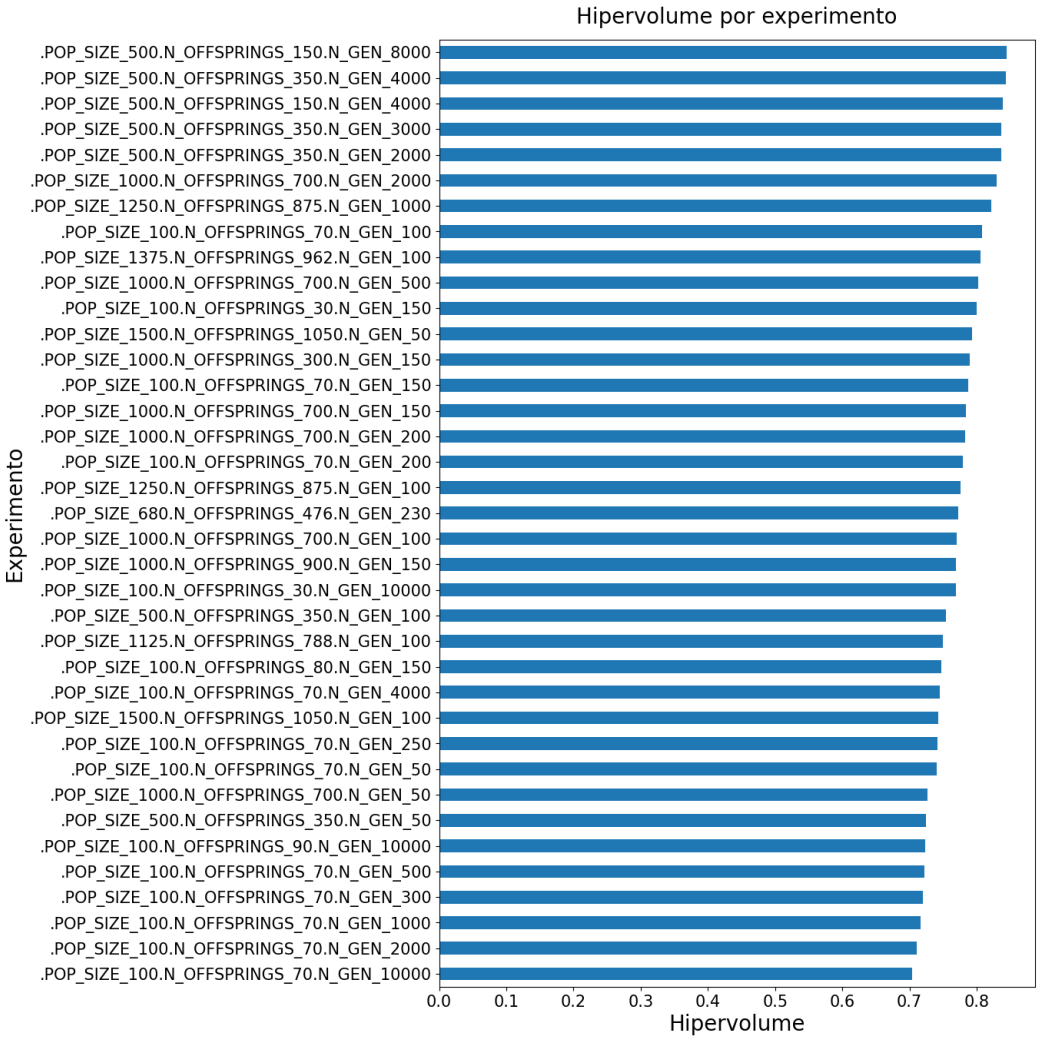
1. O tamanho da população parecia restringir o aumento da quantidade de soluções encontradas;
2. Quando a quantidade de soluções encontradas se aproximava do tamanho da população, o aumento na quantidade de gerações aparentava levar a uma melhora na qualidade das soluções.

Na coluna *Uso (%)* da Tabela 12 pode-se observar também quais experimentos alcançaram o número máximo de soluções admitido pelo experimento. Quantidade essa que é limitada pelo tamanho da população. Percebe-se que tanto os melhores quanto os piores resultados fizeram uso de todas as soluções que poderiam ser encontradas pelo experimento.

4.5.2 Hipervolumes

O gráfico na Figura 14 compara hipervolumes de todos os experimentos, que nele estão identificados por uma *string* que concatena os principais parâmetros de cada um. A análise desse gráfico sugere que até mesmo o pior experimento realizado alcançou um valor alto de hipervolume. Não passa despercebido também o fato de que é pequena a diferença entre o valor do hipervolume associado ao pior experimento com aquele obtido com o melhor experimento.

Figura 14 – Relação entre experimentos e hipervolume.



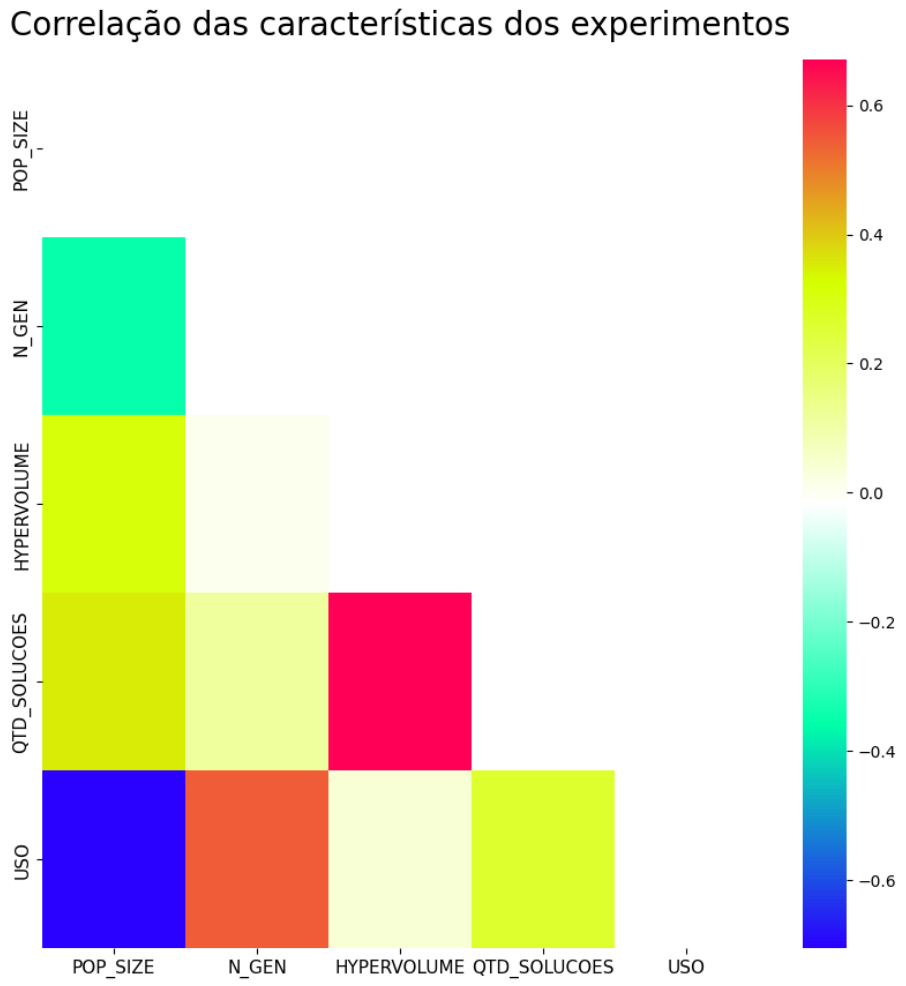
Fonte: elaborado pelo autor.

Essas interpretações sugerem que, até mesmo um resultado relativamente ruim da técnica empregada leva a uma solução muito mais adequada do que a *solução base* (Seção 4.4). Além disso, resultados obtidos de forma rápida (com apenas 150 gerações, população de 100 indivíduos e taxa de descendentes de 30%), que encontraram 30 soluções ótimas, podem realmente ter uma relevância em contextos reais, nos quais não se busca necessariamente pela melhor solução possível.

4.5.3 Correlações entre experimentos

As correlações lineares entre os experimentos podem ser observadas na Figura 15. Como a quantidade de experimentos é pequena e os próprios experimentos não foram configurados de forma aleatória, não se pode confiar plenamente nas correlações apresentadas aqui. Contudo, a análise da Figura 15 se mostra relevante para o estudo em si e para possíveis trabalhos futuros.

Figura 15 – Correlação entre as características dos experimentos.



Fonte: elaborado pelo autor.

Considerando isso, a observação do gráfico mostra que há 3 correlações significativas (mas esperadas, dada a forma como foram escolhidos os parâmetros para cada experimento). São elas: (1) a qualidade das soluções (HIPERVOLUME) cresce na medida em que aumenta-se o número de soluções encontradas (QTD_SOLUCOES), (2) o aproveitamento das soluções (USO) aumenta na medida em que aumenta-se a quantidade de gerações e (3) o aumento da população implica em diminuição do aproveitamento (USO) da quantidade de soluções disponíveis para uso.

Nesse ponto surgem algumas questões que não foram completamente respondidas com os experimentos mas que dependem de uma análise posterior, junto aos futuros usuários dessas soluções. *Qual seria um número aceitável de soluções ótimas?* A resposta a essa pergunta depende, obviamente, do próprio problema proposto e da avaliação do contexto no qual a necessidade de alocação foi identificada. Um estudo posterior poderia avaliar com mais cuidado tanto a elaboração de indicadores como a relação deles com os resultados obtidos e a percepção real daqueles que, de fato, tomam as decisões de alocação.

4.5.4 Convergência

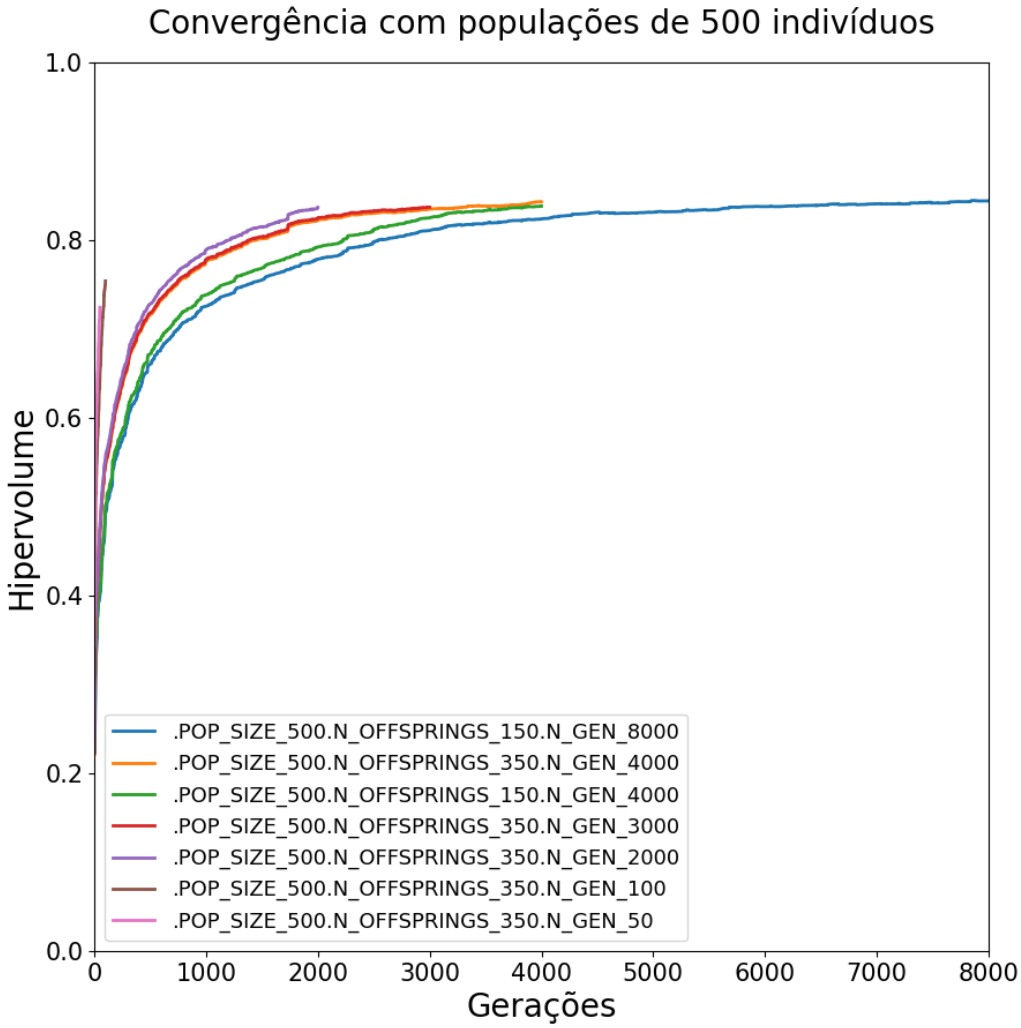
Retoma-se aqui os comentários da Seção 4.5.1, onde dissertou-se sobre os experimentos listados na Tabela 12. O melhor conjunto de soluções foi obtido em um contexto no qual avaliava-se um tamanho de população de 500 indivíduos. Variou-se tanto a quantidade de descendentes quanto o número de gerações com o objetivo de provocar o aumento no valor do hipervolume. O melhor conjunto de soluções encontrado foi obtido no último experimento realizado.

A escolha de 8.000 gerações, 30% de descendentes e 500 indivíduos é exatamente o limite admitido pela máquina utilizada para realização dos experimentos. O uso de um tamanho maior de população ou um número maior de gerações ou, ainda, de descendentes, levou à falha de execução por falta de memória. Buscou-se aqui verificar se, com uma população de tamanho médio e uma quantidade pequena de descendentes, dobrar a quantidade de gerações provocaria um aumento significativo no hipervolume das soluções encontradas.

Como pode-se perceber pela Figura 14, o aumento no hipervolume entre a melhor solução e aquela imediatamente pior não foi significativo. Nota-se aqui que, assim como existe um *trade-off* inerente às soluções encontradas para o problema proposto neste trabalho, também existe um *trade-off* entre tempo para execução das iterações do algoritmo, quantidade de memória disponível e qualidade das soluções obtidas.

Posto isso, apresenta-se o gráfico da Figura 16, que relaciona a quantidade de gerações com o hipervolume obtido, mas somente para os experimentos com população de 500 indivíduos.

Figura 16 – Convergência dos experimentos com população de 500 indivíduos.

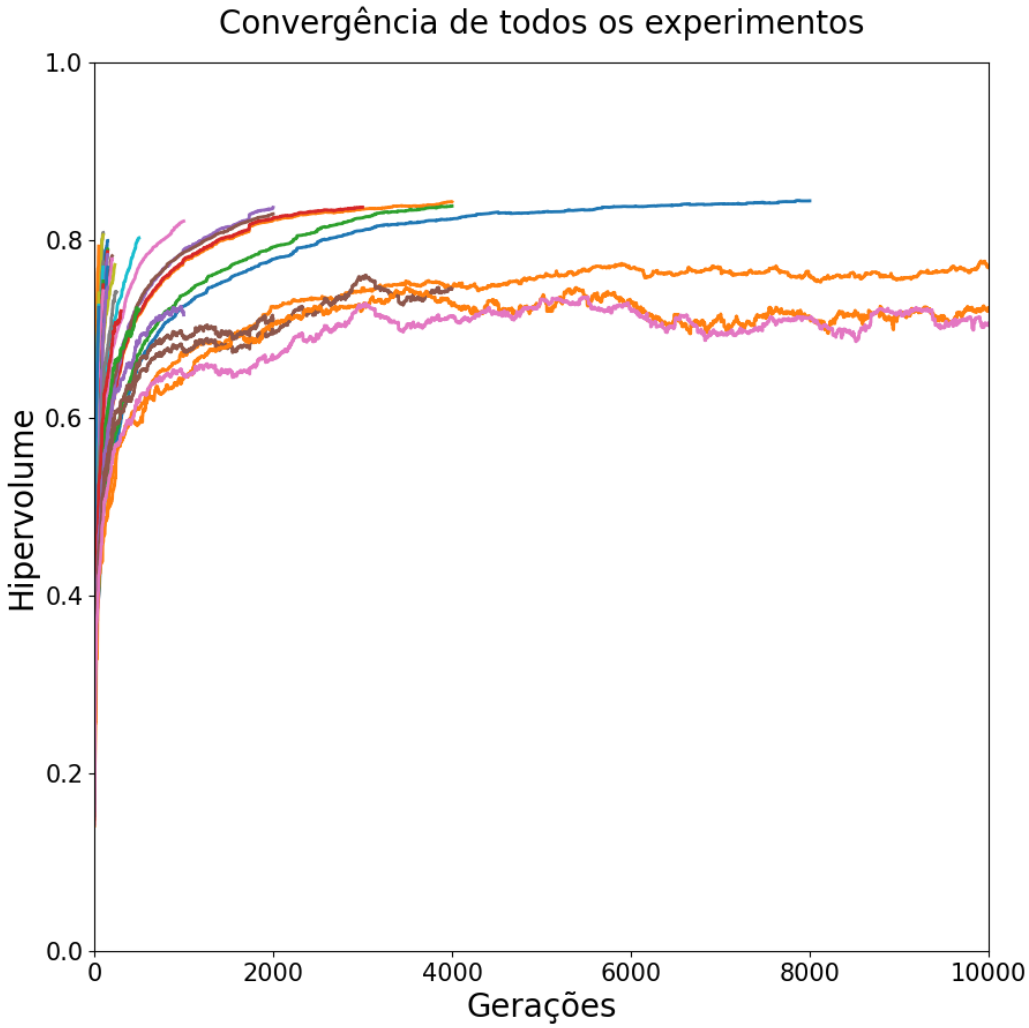


Fonte: elaborado pelo autor.

É perceptível nesse gráfico que poucas gerações não foram suficientes para que se alcançasse o maior valor de hipervolume para esta quantidade de indivíduos. Em oposição, há os experimentos com 3.000, 4.000 e 8.000 indivíduos. Esses últimos parecem oferecer pouquíssima vantagem em relação ao experimento realizado com 2.000 gerações. Considerando-se o custo da execução das gerações em relação ao benefício de um melhor hipervolume entre essas execuções, em um contexto real, provavelmente a solução com 2.000 gerações seria a mais adequada.

É relevante para a análise da convergência avaliar o comportamento de todos os experimentos realizados. Na Figura 17 são apresentadas todas as curvas associadas a eles. Esse gráfico é importante por mostrar de forma simples que nem sempre aumentar a quantidade de gerações leva a uma melhora na qualidade das soluções. Também é um gráfico que mostra que o hipervolume não necessariamente melhora *monotonamente*. Ou seja, alguns experimentos não tem curvas de convergência tão suaves e ascendentes quanto se poderia esperar.

Figura 17 – Convergência de todos os experimentos.

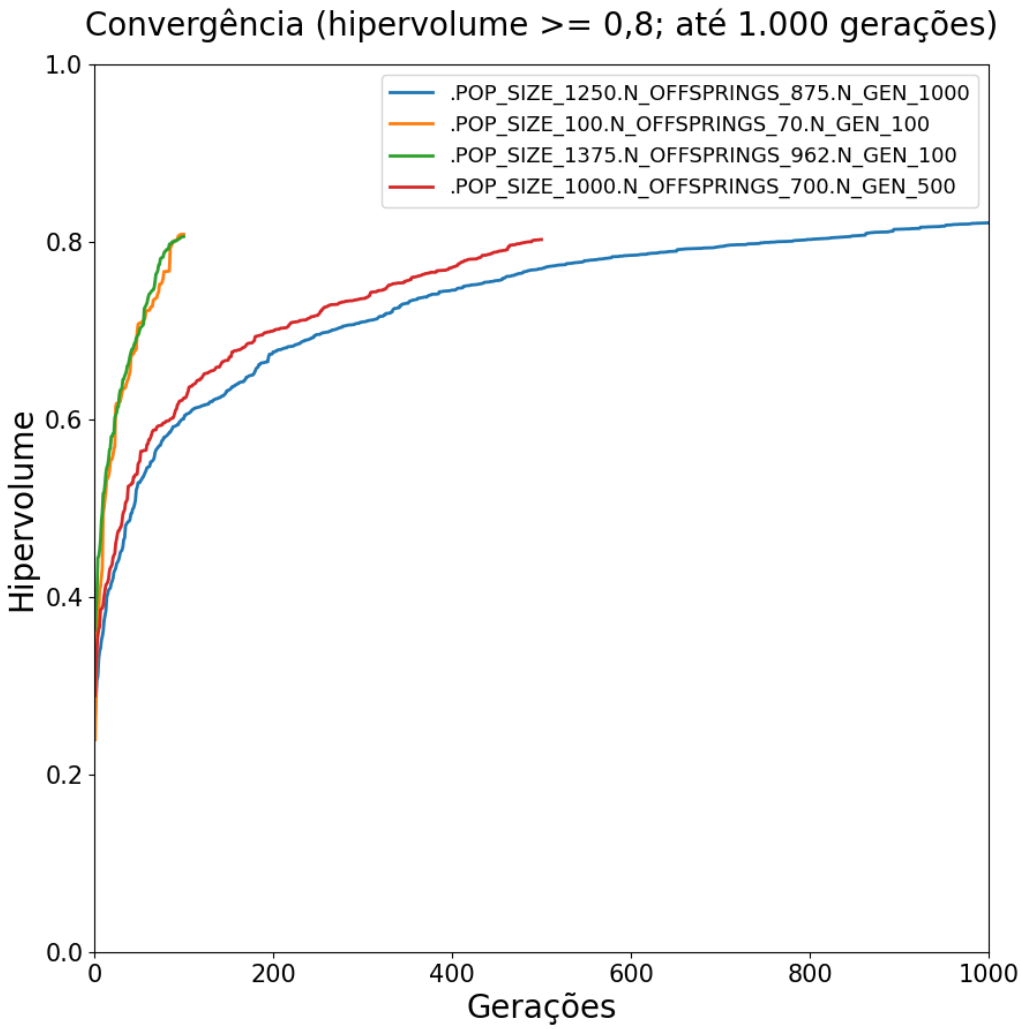


Fonte: elaborado pelo autor.

Outra situação a ser analisada por sua pertinência em um contexto real é a da Figura 18. Nessa figura foram colocadas as curvas de experimentos cuja solução tem um valor alto de hipervolume (maior ou igual a $0,8$) e um número restrito de 1.000 gerações (em comparação, há experimentos com 10.000 gerações).

Nesse gráfico da Figura 18 duas soluções se destacam por exigirem apenas 100 gerações para alcançar um hipervolume maior ou igual a $0,8$: (1) *população de 100 soluções / 70 descendentes* e (2) *população com 1375 indivíduos / 962 descendentes*. A primeira opção se destaca ainda mais, dado o tamanho reduzido da população inicial exigida por ela. Conferindo-se a Tabela 12, pode-se confirmar que esta última opção é composta por 55 combinações diferentes de desenvolvedores, que é um número relativamente baixo e, por isso, aceitável, de soluções a serem avaliadas por tomadores de decisão.

Figura 18 – Convergência dos experimentos com hipervolume maior ou igual a $0,8$ e um máximo de 1.000 gerações.



Fonte: elaborado pelo autor.

4.5.5 Experimento vencedor

A fim de ilustrar como são as soluções encontradas e permitir a comparação entre a *solução base* (Seção 4.4) e as melhores soluções de cada dimensão, na Tabela 13 são apresentadas 3 das soluções ótimas encontradas no melhor experimento realizado, aquele com o maior valor de hipervolume. Nas linhas, estão listadas as características de cada solução. As colunas ATIVIDADE, SATISFACAO e COMUNICACAO referem-se, cada uma, a uma única solução ótima. Destaca-se que, nessas soluções, a quantidade de desenvolvedores listada é exatamente a quantidade de desenvolvedores sem time definido na Tabela 10.

Quando o desenvolvedor está alocado para *INDEFINIDO*, significa que nessa solução ele não é utilizado. É o que ocorre, exemplificando, com o *Dev_98* na solução com menor indício de variabilidade de ATIVIDADE. *E por que permitir a geração de soluções nas quais há desenvolvedores não escolhidos? Se há interesse em alocação de desenvolvedores que passaram em concurso, então todos devem ter seu lugar reservado, correto?* Na verdade não. Há casos nos quais os desenvolvedores não assumem seus cargos, muitas vezes por terem passado em outros concursos, inclusive. Neste estudo essa situação foi prevista de forma deliberada com a inclusão da opção *INDEFINIDO* na geração de soluções.

Tabela 13 – Soluções com menor indício de variabilidade por dimensão avaliada.

		Soluções com menor indício de variabilidade em...		
#	**Característica**	**ATIVIDADE**	**SATISFACAO**	**COMUNICACAO**
1	SATISFACAO	1,039504	1,014605	1,041540
2	ATIVIDADE	0,435144	0,463837	0,461342
3	COMUNICACAO	0,890072	0,908020	0,836454
4	Dev_10	EntityFramework.Docs	core	tye
5	Dev_19	docfx	dotnet-api-docs	eShop
6	Dev_21	install-scripts	tye	arcade
7	Dev_26	orleans	aspire	maui
8	Dev_30	dotnet-api-docs	razor	core
9	Dev_32	tye	orleans	aspnetcore
10	Dev_34	msbuild	eShop	install-scripts
11	Dev_46	core	docs	sdk
12	Dev_51	project-system	interactive	EntityFramework.Docs
13	Dev_79	fsharp	INDEFINIDO	maintenance-packages
14	Dev_88	efcore	fsharp	dotnet-api-docs
15	Dev_89	csharplang	project-system	docfx
16	Dev_98	INDEFINIDO	INDEFINIDO	docs
17	Dev_107	maui	maui	interactive
18	Dev_121	INDEFINIDO	installer	INDEFINIDO
19	Dev_131	razor	EntityFramework.Docs	orleans
20	Dev_156	INDEFINIDO	roslyn	project-system
21	Dev_166	INDEFINIDO	docfx	razor
22	Dev_171	INDEFINIDO	maintenance-packages	msbuild
23	Dev_227	installer	INDEFINIDO	INDEFINIDO
24	Dev_232	INDEFINIDO	arcade	AspNetCore.Docs
25	Dev_275	eShop	csharplang	winforms
26	Dev_276	sdk	winforms	efcore
27	Dev_280	arcade	AspNetCore.Docs	runtime
28	Dev_292	maintenance-packages	msbuild	INDEFINIDO
29	Dev_310	interactive	runtime	INDEFINIDO
30	Dev_322	docs	sdk	roslyn
31	Dev_329	AspNetCore.Docs	INDEFINIDO	csharplang
32	Dev_343	winforms	install-scripts	installer
33	Dev_346	aspire	aspnetcore	fsharp
34	Dev_430	aspnetcore	efcore	INDEFINIDO
35	Dev_459	runtime	INDEFINIDO	INDEFINIDO
36	Dev_488	roslyn	INDEFINIDO	aspire

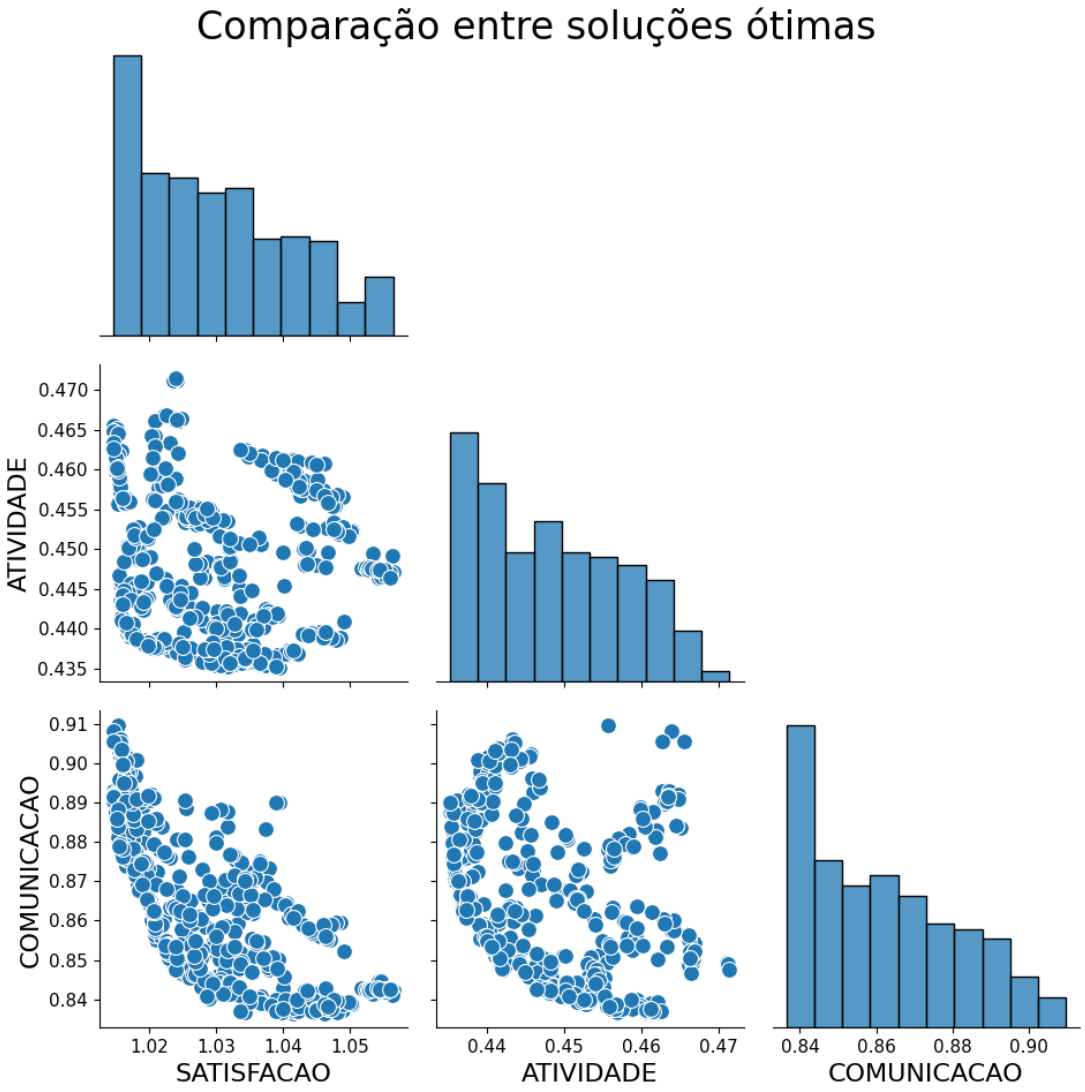
Fonte: elaborado pelo autor.

4.5.6 Análise do melhor conjunto de soluções

Explora-se aqui com mais detalhes o experimento vencedor, cuja comparação entre os resultados da função de aptidão aplicada às soluções encontradas pode ser visualizada no gráfico da Figura 19. Inicia-se a análise a partir da *diagonal principal* do gráfico, a qual

contém histogramas que evidenciam a otimização realizada. Em todos eles a distribuição sugere que a quantidade de soluções diminui na medida em que os indícios de variabilidade aumentam.

Figura 19 – Comparação entre soluções ótimas do experimento melhor avaliado.



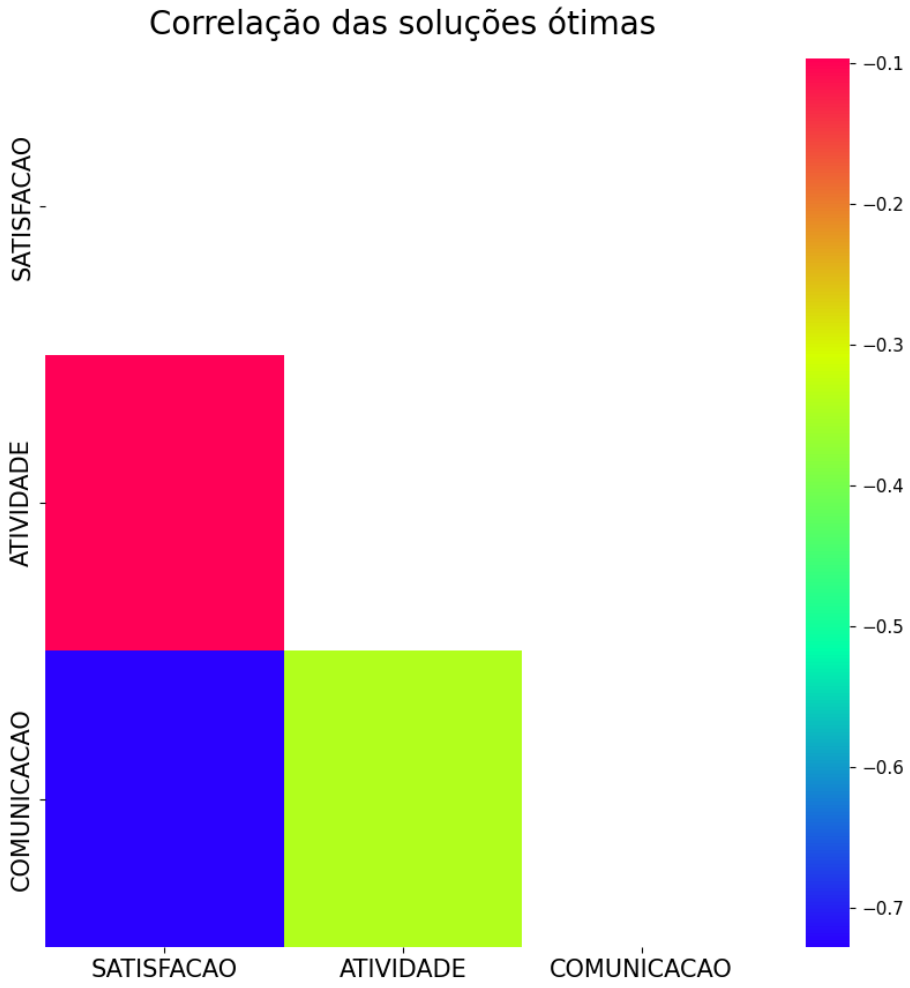
Fonte: elaborado pelo autor.

Ainda sobre a Figura 19, pode-se notar que a fronteira de Pareto estimada é mais marcante nos relacionamentos «SATISFACAO x COMUNICACAO» e «ATIVIDADE x COMUNICACAO» do que na relação «SATISFACAO x ATIVIDADE». No problema proposto, a solução *teórica* ideal seria aquela que não estivesse associada a indício algum de variabilidade para todas as 3 dimensões estimadas, visto que buscava-se minimizar os indícios de variabilidade das soluções. Percebe-se que a fronteira de Pareto estimada para «SATISFACAO x ATIVIDADE» é a que mais se aproxima dessa solução teórica.

As visões da Figura 19 não permitem extrapolações elaboradas sobre suposições em terceira dimensão, mas permitem inferências sobre como as soluções interagem entre si, par a par, em correlações lineares. Os gráficos dessa figura permitem inferir que existe um certo nível de *correlação linear negativa* entre SATISFACAO e COMUNICACAO. O mesmo não parece ser tão marcante nos outros dois relacionamentos.

Analisa-se então a Figura 20, que permite a avaliação das correlações lineares entre as soluções.

Figura 20 – Correlação entre as soluções ótimas.



Fonte: elaborado pelo autor.

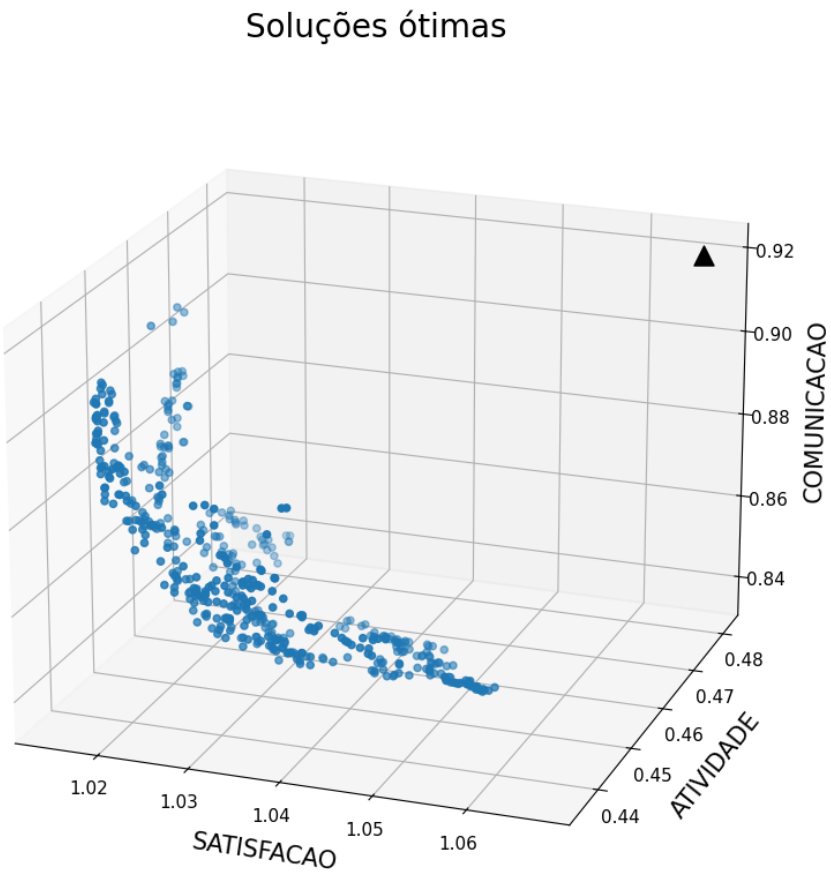
Nessa figura se destaca o fato de que, quando SATISFACAO cresce, COMUNICACAO diminui, e vice-versa. Com base no significado dessas variáveis, caso um gestor escolha uma solução cujos indícios de variabilidade de tempo envolvido com o .NET (SATISFACAO) sejam mínimos, então é bastante provável que ele precise aceitar uma solução na qual seja relativamente mais elevado o nível de indícios de variabilidade do envolvimento em *issues* (COMUNICACAO). Nessa figura também fica claro o quão baixa é a correlação entre SATISFACAO e ATIVIDADE. Sobre a correlação entre ATIVIDADE e COMUNICACAO, esta é, no máximo, moderada.

4.5.7 Visão 3D das melhores soluções

Na Seção 4.5.6, foi analisado o gráfico da Figura 19. Nesta seção apresenta-se uma visão semelhante, mas em gráficos 3D e com um novo detalhe: **a adição da *solução base* (Seção 4.4) no gráfico**, na forma de um **triângulo**.

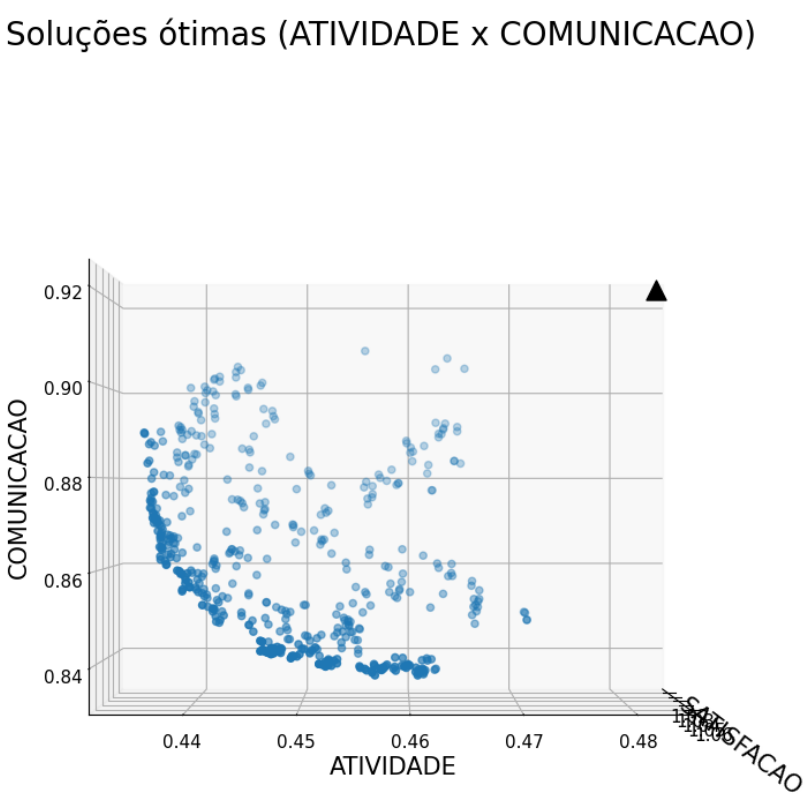
Em primeiro lugar mostra-se a Figura 21, na qual o *plano cartesiano* foi posicionado de forma a destacar a imagem tridimensional da fronteira de Pareto encontrada. Esse gráfico explicita a qualidade das soluções encontradas. As soluções parecem se afastar de forma quase que equidistante da *solução base*. Da mesma forma que se afastam da *solução base*, os pontos parecem se aproximar da *solução ideal*, onde SATISFACAO, ATIVIDADE e COMUNICACAO seriam *nulas*. As figuras 22, 23 e 24, por sua vez, mostram as visões laterais do gráfico na Figura 21.

Figura 21 – Soluções ótimas comparadas com a solução base.



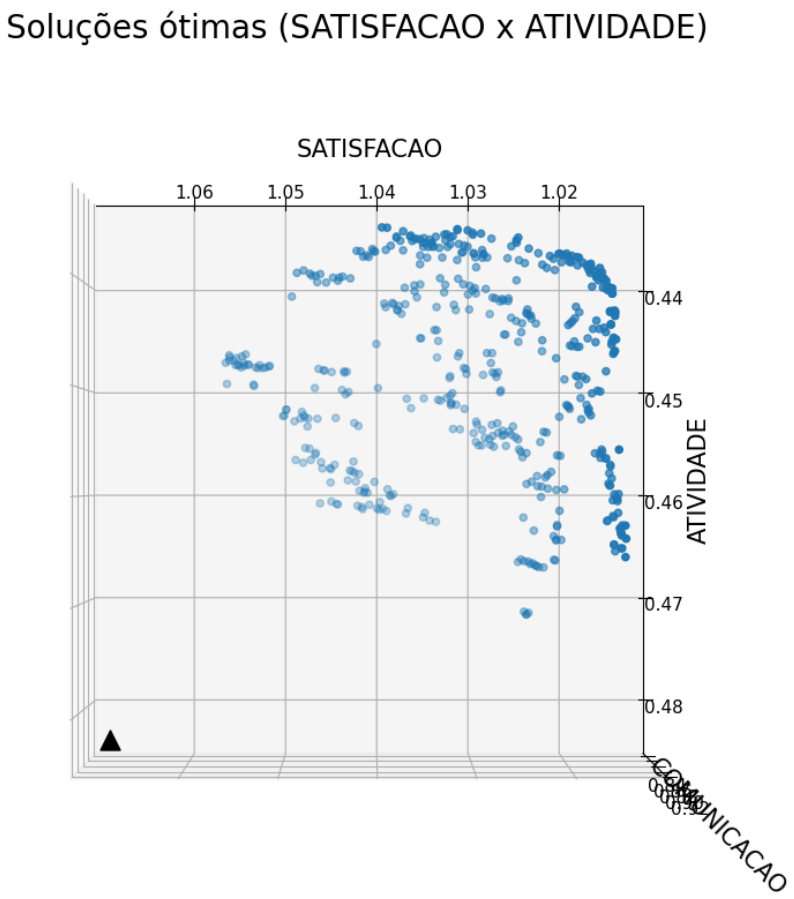
Fonte: elaborado pelo autor.

Figura 22 – Visão lateral («ATIVIDADE x COMUNICACAO»).



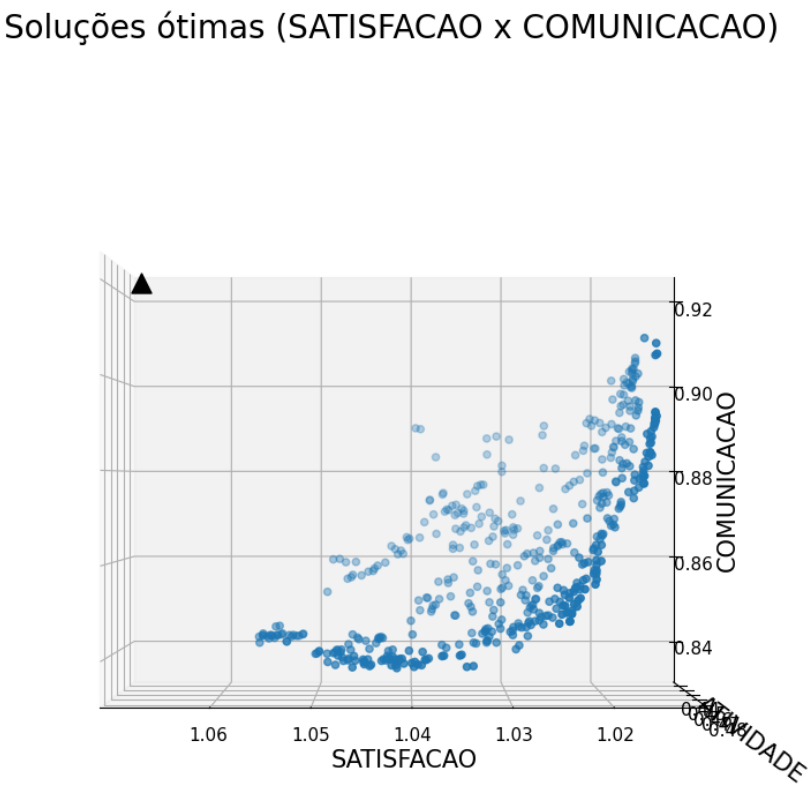
Fonte: elaborado pelo autor.

Figura 23 – Visão lateral («SATISFACAO x ATIVIDADE»).



Fonte: elaborado pelo autor.

Figura 24 – Visão lateral («SATISFACAO x COMUNICACAO»).



Fonte: elaborado pelo autor.

4.6 Sugestão de implantação

A estratégia geral sugerida para indicação de desenvolvedores para seus futuros times é a seguinte:

1. **Criar indicadores normalizados referentes às dimensões SPACE para cada um dos desenvolvedores da empresa.** Aqui o gestor precisa ter clareza dos objetivos que deseja alcançar, de forma a escolher os indicadores mais adequados para cada dimensão;

2. **Para cada time, somar os indicadores SPACE dos desenvolvedores.** Cada equipe estará associada, portanto, a (3, 4 ou 5) indicadores, que resultam da soma das estimativas das habilidades de seus desenvolvedores;

3. **Para cada dimensão SPACE, calcular os indícios de variabilidade das equipes.** O cálculo do *desvio padrão* entre as equipes, por exemplo. Esta é a *solução base*. Qualquer solução adicional proposta precisa ser melhor do que ela. Ou seja, a adição de um novo desenvolvedor, independente de qual equipe passe a ser associada ele, deve diminuir os indícios de variabilidade calculados aqui;

4. **Estimar os valores das dimensões SPACE para cada novo desenvolvedor aprovado em concurso.** O período probatório de 3 meses (Seção 2.8.1.2) pode ser aproveitado para estas estimativas. A prova final, na forma de elaboração de um *software*, por exemplo, pode ser uma fonte indispensável de informação sobre o nível de *Atividade* (talvez calculando-se a quantidade de linhas de código) e *Performance* (quantos *bugs*, erros, más-práticas, etc. foram identificadas no código);

5. **Usar a implementação sugerida neste estudo (Seção 3.3.3) para encontrar um conjunto ótimo de configurações possíveis de equipes;**

6. **Avaliar as soluções propostas de forma a verificar qual delas seria a mais adequada como escolha definitiva de configuração para os times de desenvolvimento.**

5 CONCLUSÃO

A automatização do processo de escolha do time para o qual um novo desenvolvedor será alocado é inerentemente limitada. Por mais bem elaborada que seja a automatização, as questões humanas são sempre o maior fator restritivo. A própria definição de produtividade é um desafio. Como cada pessoa tem seu próprio entendimento sobre o conceito de produtividade e sobre como ela deve ser avaliada, não se pode prescindir da discussão.

O *framework SPACE* é de grande auxílio em nortear essa discussão. Ele direciona o foco da avaliação em dimensões bem definidas, mas deixa espaço para que a interpretação de cada dimensão seja contextual. Neste estudo, por exemplo, o mais importante não foram as definições de significado de cada dimensão, mas sim, a definição do método de automatização. As dimensões SPACE, nesse caso, atuaram como encaixes genéricos o suficiente para que a empresa interessada possa usar o método aqui desenvolvido para alocar seus novos funcionários.

O método desenvolvido também parte do princípio de que será necessária uma segunda etapa de participação humana. A fronteira de Pareto encontrada exige reflexão por parte dos tomadores de decisão. O sistema indica soluções potenciais, mas são os humanos que decidem qual delas será empregada. A resposta para a questão de pesquisa, portanto, foi encontrada, mas é resultado da simbiose entre o método desenvolvido nesse trabalho e a participação humana na definição dos indicadores *SPACE* e na escolha da solução que será adotada entre as muitas que podem ser obtidas.

Dessa forma, automatizou-se sim o processo de escolha do time de um novo funcionário, como previa o objetivo geral do projeto. No entanto, a automatização foi parcial. E é justamente essa parcialidade que coloca o tomador de decisão como peça mais importante do processo de alocação, deixando a cargo da máquina o trabalho exaustivo de encontrar quais combinações de desenvolvedores são as mais promissoras. Afinal de contas, um dos objetivos gerais do projeto era justamente facilitar o trabalho de avaliação dos gestores.

O método de indicação desenvolvido é completamente voltado à otimização da produtividade do conjunto de equipes. É construído de forma a satisfazer as necessidades apresentadas pelos gestores, com uma visão macro. No entanto, ao diminuir os indícios de variabilidade entre as equipes, o próprio desenvolvedor tem ganhos práticos. Sabe-se antecipadamente que ele tem um papel na equipe e, portanto, terá seu espaço para que possa mostrar suas habilidades e se destacar. Não exatamente por ser mais habilidoso do que os demais, mas por ocupar um espaço que está praticamente reservado para alguém com o perfil dele.

Todavia, nem todos os objetivos puderam ser alcançados. Por não dispor dos dados da empresa, o estudo não abordou a inclusão das topologias fundamentais (apresentadas por Skelton, Pais and Malan (2019)), que poderão ser utilizadas em estudos posteriores. Assim como o uso de indicadores para as dimensões SPACE de *Performance* e de *Eficiência e Fluxo*. Estudos posteriores também podem ser realizados especificamente para o entendimento de quais seriam indicadores relevantes para cada uma das dimensões SPACE, mas dedicados a contextos de desenvolvimento específicos, como os domínios bancário, jurídico ou de ensino, por exemplo.

REFERÊNCIAS

AZOUZ, Y.; BOUGHACI, D. Multi-objective memetic approach for the optimal web services composition. **Expert Systems**, v. 40, n. 4, p. e13084, 2023. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.13084. Citado na página 28.

BANRISUL. **Concurso publico para o provimento de vagas para o cargo de Tecnico em Tecnologia da Informacao: edital nº 1**. 2022. Available at: https://cdn.cebraspe.org.br/concursos/banrisul_22/arquivos/ED_1_2022_BANRISUL_ABERTURA.PDF. Citado 2 vezes nas páginas 21 e 38.

Blank, J.; Deb, K. pymoo: Multi-objective optimization in python. **IEEE Access**, v. 8, p. 89497–89509, 2020. Citado na página 35.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 2018. ISSN 1677-7042. Available at: http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Citado na página 41.

DEB, K. *et al.* A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, April 2002. ISSN 1941-0026. Citado 2 vezes nas páginas 32 e 49.

FORSGREN, N. *et al.* The space of developer productivity: There's more to it than you think. **Queue**, Association for Computing Machinery, New York, NY, USA, v. 19, n. 1, p. 20–48, mar 2021. ISSN 1542-7730. Available at: https://doi.org/10.1145/3454122.3454124. Citado 4 vezes nas páginas 17, 22, 23 e 24.

GARSHASBI, S. *et al.* Optimal learning group formation: A multi-objective heuristic search strategy for enhancing inter-group homogeneity and intra-group heterogeneity. **Expert Systems with Applications**, v. 118, p. 506–521, 2019. ISSN 0957-4174. Available at: https://www.sciencedirect.com/science/article/pii/S0957417418306845. Citado 8 vezes nas páginas 25, 28, 29, 30, 31, 36, 49 e 51.

GUERREIRO, A. P.; FONSECA, C. M.; PAQUETE, L. The hypervolume indicator: Computational problems and algorithms. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 6, jul 2021. ISSN 0360-0300. Available at: https://doi.org/10.1145/3453474. Citado 2 vezes nas páginas 34 e 35.

GUO, C.; TANG, H.; NIU, B. Evolutionary state-based novel multi-objective periodic bacterial foraging optimization algorithm for data clustering. **Expert Systems**, v. 39, n. 1, p. e12812, 2022. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12812. Citado na página 28.

HOSSEINI, M.; AKHAVAN, P. A model for project team formation in complex engineering projects under uncertainty: A knowledge-sharing approach. **Kybernetes**, v. 46, p. 00–00, 06 2017. Citado na página 28.

LIU, C. **A floresta sombria**. [*S.l.: s.n.*]: Suma, 2017. (O problema dos três corpos). ISBN 9788554510060. Citado na página 7.

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. [*S.l.: s.n.*]: Editora Atlas S.A., 2021. ISBN 9788597026566. Citado na página 45.

SHAO, Y.; GEYER, P.; LANG, W. Integrating requirement analysis and multi-objective optimization for office building energy retrofit strategies. **Energy and Buildings**, v. 82, p. 356–368, 2014. ISSN 0378-7788. Available at: https://www.sciencedirect.com/science/article/pii/S0378778814005684. Citado 3 vezes nas páginas 28, 29 e 30.

SHUKLA, S. *et al.* Multi-objective cross-version defect prediction. **Soft Computing**, v. 22, n. 6, p. 1959–1980, Mar 2018. ISSN 1433-7479. Available at: https://doi.org/10.1007/s00500-016-2456-8. Citado 3 vezes nas páginas 28, 29 e 30.

SI, G. *et al.* A reliability-and-cost-based framework to optimize maintenance planning and diverse-skilled technician routing for geographically distributed systems. **Reliability Engineering System Safety**, v. 226, p. 108652, 2022. ISSN 0951-8320. Available at: https://www.sciencedirect.com/science/article/pii/S0951832022002873. Citado 4 vezes nas páginas 28, 29, 30 e 31.

SKELTON, M.; PAIS, M.; MALAN, R. **Team Topologies: Organizing Business and Technology Teams for Fast Flow**. IT Revolution Press, 2019. ISBN 9781942788829. Available at: https://books.google.com.br/books?id=Pj-IDwAAQBAJ. Citado na página 84.

SORDI, J. O. D. **Elaboração de pesquisa científica**. [*S.l.: s.n.*]: Saraiva Educação S.A., 2013. ISBN 9788502210325. Citado na página 45.

THIEL, P. **De zero a um: O que aprender sobre empreendedorismo com o Vale do Silício**. Objetiva, 2014. ISBN 9788539006373. Available at: https://books.google.com.br/books?id=EtmxBAAAQBAJ. Citado na página 21.

TICONA WALDO GONZALO CANCINO E DELBéM, A. C. B. Algoritmos evolutivos para otimização multi-objetivo. 2008. Available at: https://web.icmc.usp.br/SCATUSU/RT/Notas_Didaticas/nd_76.pdf. Citado 4 vezes nas páginas 25, 32, 33 e 34.

VELDHUIZEN, D. A. V.; LAMONT, G. B. *et al.* Evolutionary computation and convergence to a pareto front. *In*: CITESEER. **Late breaking papers at the genetic programming 1998 conference**. [*S.l.: s.n.*], 1998. p. 221–228. Citado na página 26.

ZHANG, L.; ZHANG, X. Multi-objective team formation optimization for new product development. **Computers Industrial Engineering**, v. 64, n. 3, p. 804–811, 2013. ISSN 0360-8352. Available at: https://www.sciencedirect.com/science/article/pii/S0360835212003294. Citado 4 vezes nas páginas 28, 29, 30 e 31.

ZHANG, Z.; MA, S.; JIANG, X. Research on multi-objective multi-robot task allocation by lin–kernighan–helsgaun guided evolutionary algorithms. **Mathematics**, v. 10, n. 24, 2022. ISSN 2227-7390. Available at: https://www.mdpi.com/2227-7390/10/24/4714. Citado 3 vezes nas páginas 28, 29 e 30.

ZHAO, H. *et al.* Multi-objective optimization for football team member selection. **IEEE Access**, v. 9, p. 90475–90487, 2021. ISSN 2169-3536. Citado 4 vezes nas páginas 28, 29, 30 e 31.