Beyond Subjectiviness: Assessing Abilities and Preferences to Create Software Development Teams

Lucas Alves
Pontifical Catholic University of Minas Gerais
Belo Horizonte, Brazil
lucas.alves.696400@sga.pucminas.br

Vinícius Ricardo
Pontifical Catholic University of Minas Gerais
Belo Horizonte, Brazil
vinicius.ricardo@sga.pucminas.br

Laerte Xavier
Pontifical Catholic University of Minas Gerais
Belo Horizonte, Brazil
laertexavier@pucminas.br

ABSTRACT

The creation of software development teams that are affected by performance issues is a problem frequently observed in companies in the software development market. This process is commonly done through subjective methodologies. Such methodologies can be influenced by interpersonal relationships and susceptible to human error. This paper proposes a quantitative and data-oriented alternative to the process of forming workgroups through the use of a genetic algorithm capable of optimizing collaborator's abilities and preferences when executing a specific task within a project. As a result, we show that the use of such genetic algorithm is able to create teams similar to the teams assembled by the project managers of companies in the industry of software engineering. Therefore, the ability of genetic algorithm on supporting the process of develoment teams assembly becomes evident.

CCS CONCEPTS

• **Software and its engineering** → **Software evolution**.

ACM Reference Format:
Lucas Alves, Vinícius Ricardo, and Laerte Xavier. 2021. Beyond Subjectiviness: Assessing Abilities and Preferences to Create Software Development Teams. In *1st Brazilian Workshop on Intelligent Software Engineering (ISE'21), September 27th, 2021, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.5753/ise.2021.17276

1 INTRODUÇÃO

Apesar da movimentação crescente para a tomada de decisões orientadas a dados, o processo de seleção de membros para uma equipe de trabalho ainda é essencialmente subjetivo. Líderes de equipe utilizam sua percepção das habilidades de um colaborador para alocar recursos à determinadas tarefas. Assim, esse processo pode ser suscetível ao erro humano ou enviesado por relacionamentos interpessoais, gerando equipes que apresentam resultados aquém do esperado [1]. Nesse contexto, a aplicação de abordagens estudadas na *Search Based Software Engineering* (SBSE) [6, 7, 10] representa uma possível alternativa para tais metodologias subjetivas. No entanto, os melhores critérios e metodologias para formação de uma equipe de trabalho eficiente ainda não são claros. Isto é, ainda não é possível prever se um determinado método ou processo irá de fato gerar equipes de desenvolvimento eficientes. Portanto, **o problema que este trabalho endereça refere-se à subjetividade do processo de formação de equipes de desenvolvimento de software**.

Nesse contexto, o problema investigado nesta pesquisa é relevante na medida em que se observa que equipes de desenvolvimento apresentam resultados insatisfatórios quando não possuem os fatores técnicos ou motivacionais necessários para executar uma determinada tarefa. Assim, o processo de seleção, gestão e alocação de desenvolvedores em tarefas compatíveis torna-se crucial para a engenharia de software no que tange a perspectiva de fatores humanos e pessoais [9]. Portanto, é necessário que abordagens mais claras sejam identificadas para auxiliar no desenvolvimento de ferramentas que orientem esse processo.

Diante do exposto, este trabalho tem como objetivo geral **realizar uma avaliação de um método de otimização para montagem de equipes de trabalho, buscando identificar a viabilidade de utilização de uma metodologia quantitativa baseadas em habilidades mineradas do GitHub**. Para tanto, realiza-se um estudo de caso baseado na análise de um modelo de otimização orientado a dados em comparação com os processos subjetivos comuns no contexto de desenvolvimento de software de uma grande empresa da cidade de Belo Horizonte. Particularmente, busca-se responder às seguintes questões de pesquisa:

RQ1. Algoritmos de otimização podem auxiliar a tomar decisões no processo de montagem de equipes?
RQ2. Qual a adequação de equipes geradas por tais algoritmos?

Como contribuição, espera-se obter uma maior compreensão do potencial da aplicação de meta-heurísticas de otimização na solução de problemas de engenharia de software, especificamente na montagem de equipes de trabalho.

Estrutura do artigo. Na Seção 2 é descrita a metodologia proposta. Na Seção 3 são apresentados os resultados obtidos e as questões de pesquisa propostas são respondidas. Na Seção 4, implicações dos resultados obtidos. Na Seção 5, são discutidos trabalhos relacionados. Por fim, nas Seções 6 e 7 são discutidas ameaças à validade e são apresentadas as conclusões finais.

2 METODOLOGIA

A metodologia do presente trabalho se configura como explicativa, pois procura-se identificar os fatores que influenciam no processo de formação de equipes de software, analisando o processo e avaliando um método para sua otimização. Os dados utilizados para realização deste estudo foram coletados em uma empresa de tecnologia

ISE'21, September 27th, 2021, Brazil

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.5753/ise.2021.17276

de Belo Horizonte. Fundada em 2004, a empresa atua no ramo de hospedagem e distribuição de vídeos online, desenvolvendo tanto produtos relacionados a esta demanda quanto projetos customizados. Atualmente ela conta com aproximadamente 120 colaboradores. O restante da metodologia é descrita através dos seguintes passos:

1. Implementação do algoritmo genético. Neste trabalho adota-se um modelo de otimização proposto por Ferreira et al. [3] que também busca aperfeiçoar a formação de equipes de trabalho de acordo com as habilidades e as preferências de cada integrante de um determinado projeto. O modelo proposto é construído a partir de uma série de matrizes de atribuição de valores de representação do relacionamento Atividade x Habilidade requerida, Pessoa x Habilidade e Pessoa x Atividade (preferência). O algoritmo também propõe a atribuição de valores *alfa* e *beta* para determinar pesos à habilidade e preferência, respectivamente. Para a montagem das equipes, o algoritmo calcula o total das habilidades do candidato, somado à sua preferência. Esse cálculo, que representa o *fitness* de cada indivíduo da população, pode ser observado na Equação 1.

$$C = \sum_{h=1}^{H} \sum_{n=1}^{N} A_{hn}\beta + \sum_{n=1}^{N} Pr_n\alpha \quad (1)$$

Onde C é o candidato ótimo, H representa a habilidade necessária para realização da atividade, N o número de participantes, A_{hn} a habilidade h do participante n e $\mathrm{Pr_n}$ a preferência do participante n pela atividade. As variáveis α e β são fatores de relevância utilizados para definir a importância de cada parâmetro na função objetivo.

Assim, o algoritmo desenvolvido em Python requer duas entradas para sua execução. A primeira entrada é uma matriz representativa das atividades a serem realizadas ao longo do desenvolvimento do projeto (eixo X) e as habilidades necessárias para realização dessa atividade (eixo Y). O eixo de atividades foi criado de forma empírica com base no histórico da empresa. Por outro lado, complementando o trabalho de Ferreira et al. [3], o eixo de habilidades foi montado com base nos dados dos desenvolvedores da empresa, minerados no GitHub. Por fim, a segunda entrada define o número máximo de membros da equipe. Esse valor representa o número de cromossomos de cada indivíduo dentro da população do algoritmo.

As etapas de execução do processo são apresentadas no Algoritmo 1. Ao iniciar a execução do algoritmo, uma população de 200 indivíduos é inicializada (Linha 1). Cada indivíduo desta população representa uma configuração possível de equipe de trabalho e seus cromossomos representam os membros de cada uma dessas configurações. Cada geração do algoritmo é composta por três passos básicos. Primeiro, na Linha 5, os indivíduos são avaliados com base na função objetivo (Fórmula 1). Dessa forma, o seu *fitness* é estabelecido. Em seguida, o indivíduo com o maior *fitness* é registrado (Linha 8). Por fim, um processo de cruzamento e mutação é realizado sobre a população para que novos indivíduos sejam gerados dentro do espaço de busca (Linhas 10 e 11). Esse processo é repetido por 10.000 gerações. Os valores iniciais de população e o número de gerações foram estabelecidos com base em testes preliminares de execução do algoritmo, utilizando entradas aleatórias dentro do espaço de busca.

Como saída, o algoritmo fornece um vetor com os membros da equipe escolhidos após a execução da otimização. Tal vetor pode ter um número de membros menor ou igual ao número máximo definido na entrada. A saída do algoritmo representa a configuração ótima de uma equipe para execução das atividades.

Algoritmo 1 Algoritmo de Otimização para Montagem de Equipes

```
1:  inicializar_populacao()
2:  melhor_individuo_global = Null
3:  for Cada iteração do
4:      for Cada indivíduo da população do
5:          individuo.calcular_fitness()
6:      end for
7:      if individuo.fitenss > melhor_individuo_global.fitness
        then
8:          melhor_individuo_global = individuo
9:      end if
10:     realizar_cruzamento()
11:     realizar_mutacao()
12: end for
13: return melhor_individuo_global
```

2. Obtenção dos dados de habilidade. As métricas de habilidade foram obtidas a partir da análise histórica de atividades de cada desenvolvedor da empresa participante. O cálculo da habilidade é apresentado na Equação 2, onde *CO* representa o número de *Commits* realizados pelo desenvolvedor; *CR*, o número de *Code reviews*, ou revisões de código; *I*, o número de *Issues* abertas ou fechadas; *PR*, o número de *Pull Requests* abertos e aceitos; *Lines of code* (*LOC*), o número de linhas de código geradas pelo desenvolvedor nos *commits*. A métrica de *LOC* foi adotada nesse contexto uma vez que os valores resultantes são utilizados apenas para comparar desenvolvedores no escopo de uma mesma habilidade (i.e., não são realizadas comparações entre diferentes linguagens ou tecnologias). Cada valor de LOC produzido pelo programador é multiplicado pelo valor *DMM* do *commit* associado. O valor resultante da métrica *DMM* é calculado pela biblioteca *pydriller*.[1] O valor da métrica varia entre 0,0 (todas as alterações são arriscadas) a 1,0 (todas as alterações são de baixo risco) [8]. Por fim, os valores foram normalizados para que o valor de Habilidade H esteja entre 0 e 10, facilitando sua utilização pelo Algoritmo Genético.

$$H = CO + CR + I + PR + (LOC * DMM) \quad (2)$$

A mineração das métricas de habilidade (H) foram realizadas a partir de um script implementado em *Python* capaz de processar os dados de colaboradores dos repositórios da empresa participante. Primeiramente, foi obtida a lista de todos os repositórios da organização a partir de um conjunto de chamadas à *API* (*Application Programming Interface*) do *GitHub*. Esses repositórios, em seguida, foram classificados em relação à principal linguagem de programação utilizada e demais tecnologias relevantes para este trabalho. Projetos relacionados a *frameworks* Javascript específicos foram identificados a partir da presença deles nos arquivos de dependência `package.json`. Os repositórios representativos das habilidades de DevOps foram selecionados manualmente uma vez que, devido a uma peculiaridade do processo de trabalho do time responsável, todas as atividades são concentradas em um único conjunto de repositórios.

[1] https://pypi.org/project/PyDriller/

Após a mineração do conjunto de repositórios da empresa, foram contabilizadas as contribuições de cada desenvolvedor. A biblioteca *pydriller* também foi utilizada para auxiliar nesse processo. Os dados foram agregados em relação às tecnologias relevantes do repositório e associados aos respectivos colaboradores. Por fim, foi realizada uma primeira validação dos dados gerados. Observou-se que, devido a uma peculiaridade da plataforma, nem todas as contribuições de um colaborador foram quantificadas corretamente, já que uma contribuição pode ser identificada pelo *login*, e-mail ou nome do desenvolvedor. Dessa forma, uma nova agregação de dados foi realizada de forma manual para que todos os possíveis registros de um desenvolvedor fossem associados de forma correta. Portanto, para completa automação do processo, destaca-se a necessidade da padronização dos dados de contribuição de todos os desenvolvedores do projeto.

3. Obtenção dos dados de preferências. As métricas de preferência, por sua vez, foram obtidas a partir de um questionário com os próprios desenvolvedores. Nesse caso, foi solicitado que cada desenvolvedor avaliasse sua preferência de atuação em cada uma das tecnologias analisadas, atribuindo à mesma um valor de 1 a 10. Tanto as métricas de habilidade quanto as de preferência foram selecionadas de acordo com as tecnologias utilizadas pela empresa. Assim, esses agrupamentos possibilitaram a separação das diversas habilidades necessárias para criação de cenários hipotéticos que foram utilizados na avaliação da técnica. As habilidades e preferências avaliadas são relacionadas às seguintes tectonologias: Java, Python, NodeJS, PHP, ReactJS, NextJS e DevOps.

4. Análise dos resultados. Como forma de avaliar os resultados obtidos com a execução do algoritmo, foram definidos cenários hipotéticos, baseados em projetos históricos desenvolvidos pela própria empresa. Todos os cenários são caracterizados como sistemas web, com pelo menos uma linguagem *back-end* e outra *front-end*. Também se supõe que todo projeto deve ter pelo menos uma pessoa com conhecimentos de DevOps. Na abordagem proposta a criação de uma equipe é independente da outra, logo um profissional pode estar presente em mais de um time. Para cada projeto, foi realizada uma comparação entre as equipes montadas pelo algoritmo genético e as montadas pelos líderes técnicos da empresa. A caracterização desses projetos pode ser observada abaixo:

Projeto 1: *Back-end* em Java e *front-end* em ReactJS.
Projeto 2: *Back-end* em Java e *front-end* em NextJS.
Projeto 3: *Back-end* em Python e *front-end* em ReactJS.
Projeto 4: *Back-end* em PHP e *front-end* em NextJS.
Projeto 5: *Back-end* em NodeJS e *front-end* em ReactJS.
Projeto 6: *Back-end* híbrido com Java e Python e *front-end* em ReactJS.

Em entrevista com dois líderes da empresa participante, foi solicitado que escolhessem, entre os participantes do estudo, aqueles que fossem os mais aptos para realização de cada projeto hipotético. Em seguida, utilizando a fórmula proposta no estudo, foi calculado o *fitness* total de cada equipe formada pelos gestores de projetos, para que então elas sejam comparadas com as equipes geradas via algoritmo genético. Por fim, foi enviado um formulário para os mesmos líderes participantes, onde foram apresentados os times gerados pelo algoritmo genético para cada projeto hipotético, de forma que pudessem ser comparados com o time escolhido pelo líder para realização do mesmo projeto.

3 RESULTADOS

RQ1: Algoritmos de otimização podem ajudar a tomar decisões no processo de montagem de equipes?

Para avaliar se o algoritmo de otimização estudado é capaz de auxiliar os líderes no processo de formação de equipes, foram realizadas análises entre as equipes geradas pelo algoritmo e as equipes formadas pelos líderes de projeto. Antes de iniciar o processo de formação das equipes, foi primeiro necessário avaliar as preferências e habilidades técnicas dos desenvolvedores participantes do estudo. Esses dados, assim como as configurações do projeto e o número de membros do time, são fornecidos como entrada para o algoritmo de otimização e foram também fornecidos para que os líderes utilizassem como base no processo de montagem de seus times.

A Figura 1 apresenta os resultados obtidos para as preferências dos desenvolvedores por tecnologia. Analisando os dados, pode ser observado que, com exceção de NodeJs, os desenvolvedores da empresa não se interessam em trabalhar com as tecnologias *Back-end* utilizadas nos projetos da empresa, com Java, Python e PHP apresentando notas menores que cinco para mais de 50% das avaliações dos participantes. DevOps também recebeu um índice de rejeição elevado (nota de no máximo 5 para 60% dos desenvolvedores). Entretanto, observa-se que no momento em que o estudo foi realizado, a empresa possuía apenas quatro profissionais atuando na área. Por outro lado, as tecnologias *front-end* aparentam ser bem recebidas pelos participantes. Um total de 36 desenvolvedores responderam o formulário de preferências.

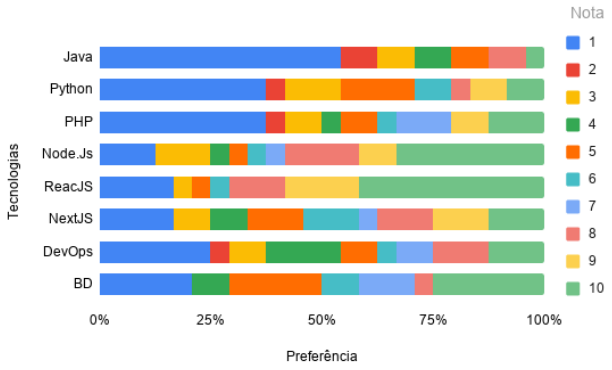

Figure 1: Preferência dos desenvolvedores por tecnologia.

Na Figura 2 observa-se a distribuição das habilidades técnicas dos desenvolvedores em cada tecnologia. O eixo X do gráfico representa as habilidades avaliadas dos desenvolvedores, enquanto o eixo Y indica, de 1 a 10, a nota que os desenvolvedores receberam para cada tecnologia, com base na mineração de informações do repositórios dos projetos da empresa. Dessa forma, considerando todos os elementos analisados, observa-se que todas as tecnologias, com exceção de ReactJs, apresentaram medianas igual a zero. ReactJs apresentou uma mediana igual a 1. Em diversos casos, desenvolvedores possuíam contribuições em apenas uma ou duas

linguagens, contribuindo para um elevado número de zeros no *dataset*. Além disso, o gráfico apresentado na Figura 2 demonstra que a habilidade mais comum entre os participantes é ReactJs, enquanto as mais escassas são Java, Python e DevOps, cujos primeiro quartil, mediana e terceiro quartil são iguais a zero.

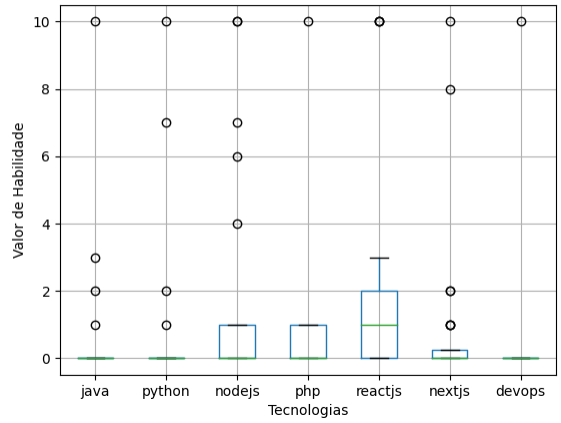

Figure 2: Habilidade dos desenvolvedores por tecnologia.

Com os dados de habilidades e preferência obtidos, o algoritmo foi executado de forma a gerar uma equipe para cada um dos projetos propostos. Após a formação dos times, foi calculado o *fitness* das equipes construídas pelos líderes para os cenários hipotéticos propostos de forma que pudessem ser comparadas com as equipes geradas pelo algoritmo. Os resultados da comparação entre eles são representados na Figura 3. Nela, pode ser observado o *fitness* calculado das equipes formadas para cada projeto. AG identifica as equipes estipuladas via algoritmo genético, enquanto L1 e L2 indicam as equipes escolhidas por cada um dos líderes. O eixo Y aponta o *fitness* alcançado por cada time.

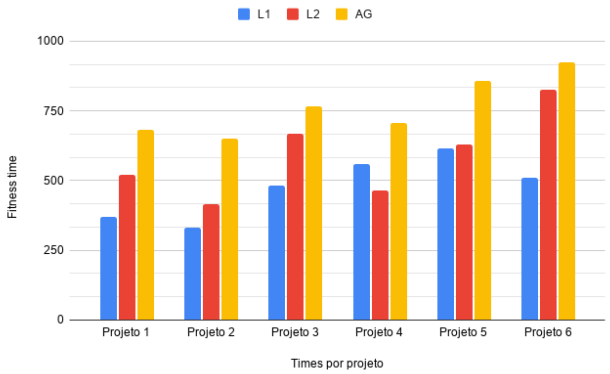

Figure 3: Comparação da fitness de cada time por projeto.

Para o Projeto 1, observa-se que o time formado por L1 teve um *fitness* total de 371,35, enquanto os times de L2 e AG obtiveram 521,85 e 683,15, respectivamente. Para o Projeto 2, L1 obteve o valor 329,85; L2, 415,4; e AG, 652,15. Nos cenários 3, 4 e 5, que são os projetos que envolvem as habilidades mais comuns (Figura 2) e as tecnologias mais populares entre os desenvolvedores (Figura 1), observa-se que as avaliações dos times formados pelos líderes aproximou-se consideravelmente dos times formados pelo algoritmo (com uma diferença de, no máximo, 200). Assim L1 obteve *fitness* igual a 482,05; L2, 669,35; e AG, 765,45 no Projeto 3. No Projeto 4, L1 e L2 foram avaliados com um *fitness* de 558,25 e 466,2, respectivamente, enquanto AG foi avaliado com um *fitness* de 705,8. Os times formados por L1, L2 e AG receberam respectivamente as notas 615,05, 630,65 e 856,05 no Projeto 5. Por fim, os coeficientes de *fitness* 509,35, 827,1, 922,95 foram obtidos para as equipes de L1, L2 e AG no Projeto 6.

Observando os dados apresentados na Figura 3, é possível identificar quão próximo os valores obtidos por ambos os líderes ficaram dos valores obtidos pelo AG nos cenários que são mais abundantes as preferências e habilidades dos participantes nas tecnologias abordadas. Casos como esses indicam que o algoritmo é capaz de simular times que são similares aos propostos por líderes de projeto. Entretanto, observa-se ainda que o *fitness*, em todos os casos, é maior para AG do que para os líderes L1 e L2. Nos cenários onde ocorreu uma elevada disparidade entre os valores obtidos entre líderes e algoritmo, observa-se que a tecnologia é rejeitada pelos participantes, ou poucos têm conhecimento nela. Dessa forma, tem-se uma maior dificuldade para o líder encontrar o profissional mais qualificado para o trabalho. Em casos como esses, o algoritmo pode auxiliar o líder a encontrar o melhor profissional para o caso.

Portanto, em cenários onde o número de profissionais que têm expertise e afinidade com as tecnologias trabalhadas é elevado, o algoritmo apresenta números similares ao dos líderes, mostrando sua capacidade de montar times compatíveis com os montados pelos líderes de grandes empresas do mercado. Em casos de disparidade entre os valores das equipes formadas pelos líderes e pelo algoritmo, as equipes formadas pelo algoritmo apresentaram valores superiores de *fitness* em todos os casos. Dessa forma, o algoritmo se mostrou capaz de auxiliar o líder na identificação de profissionais preparados para o trabalho utilizando métricas objetivas e provendo, assim, um embasamento alternativo às metodologias subjetivas.

RQ2: Qual a adequação de equipes geradas por algoritmos genéticos?

Com o propósito de avaliar a adequação das equipes geradas via algoritmo, foi solicitado que os líderes avaliassem os times formados, com uma nota de 1 a 5, comparando-os com aquelas que já haviam apontado como mais indicadas para cada projeto. São apresentadas na Tabela 1 as notas apontadas por L1 e L2 para cada um dos times formados pelo algoritmo. Observa-se que, dos times gerados pelo algoritmo, apenas AG 1 e AG 6 receberam notas inferiores a quatro de ambos os líderes. Quando questionados sobre as notas dadas para esses dois times, os líderes indicaram que, segundo seus conhecimentos acerca das pessoas escolhidas, os times não teriam ficado equilibrados em questão de preferências/conhecimentos. Por exemplo, o seguinte fragmento ilustra a justificativa de L2 para a nota dada ao Time AG 1: *Considerando o que eu conheço dessas pessoas, há muitos front-ends e apenas uma pessoa que eu imagino ter conhecimento em Java, ou que goste dessa tecnologia. Para montar o time eu tentaria colocar ao menos duas pessoas no back-end e duas no front-end.* (L2)

Table 1: Avaliação de eficiência dos times gerados via algoritmo.

	L1	L2
Time AG 1	3	3
Time AG 2	3	4
Time AG 3	2	5
Time AG 4	5	5
Time AG 5	5	5
Time AG 6	3	3

Ainda na Tabela 1, observa-se que os Times AG 4 e AG 5 receberam a nota máxima (nota 5) de ambos os líderes. Note que esses times obtiveram *fitness* total próximo das equipes formadas pelos líderes (Figura 3). Quando questionados sobre as notas dadas para essas duas equipes, as respostas apontaram que as equipes eram consistentes e bem equilibradas. A outra equipe com avaliação similar a dos líderes, Time AG 3, recebeu notas divergentes: enquanto L2 deu a nota máxima, considerando a equipe bem equilibrada, L1 deu a menor nota entre todas as equipes avaliadas (nota 2). A justificativa foi que, entre os desenvolvedores escolhidos pelo algoritmo, a maioria não possuía proficiência nas tecnologias exigidas. Por exemplo, o seguinte comentário ilustra a justificativa de L1 sobre a nota dada para o Time AG 3: *Não posso falar sobre o Dev18, mas os outros desenvolvedores não trabalham com essas tecnologias ou não possuem proficiência na área de atuação.* (L1)

Quando perguntados sobre os critérios utilizados durante a formação de suas próprias equipes, os líderes apontaram que avaliaram as preferências, experiência e expertise dos profissionais nas linguagens exigidas em cada projeto, variáveis que nossa função de *fitness* também leva em consideração ao calcular o *fitness* total de cada participante. Por exemplo, a seguinte resposta foi dada pelo líder L2 quando perguntado sobre quais critérios levou em consideração ao formar suas equipes: *A preferência e experiência com as linguagens exigidas, e a expertise dos profissionais.* (L2)

4 IMPLICAÇÕES

Implicação #1: O algoritmo apresentou um resultado avaliado como positivo por ambos os líderes em todos os projetos propostos. Dessa forma, observa-se que a abordagem avaliada para gerar equipes de trabalho poderia ser utilizada em ambientes análogos aos de uma empresa de desenvolvimento de software. Como exemplo, um líder de equipe novo na empresa, e que ainda não possui conhecimento de todo o contexto da equipe de desenvolvedores, pode utilizar as recomendações apresentadas pelo algoritmo para orientar seu processo de escolha. Ao mesmo tempo, líderes mais experientes podem utilizá-lo para validar a escolha dos membros de um determinado projeto.

Implicação #2: Os resultados apresentados pelas avaliações dos líderes acerca das equipes geradas foi satisfatório, tendo uma nota média de 3,83. Em apenas um dos casos as equipes foram avaliadas com uma nota abaixo de 3. Portanto, mesmo em casos onde a avaliação foi pior, as equipes ainda podem ser utilizadas como base para a seleção de equipes. Ou seja, nem os piores casos de equipes geradas devem ser descartados como equipes inviáveis. Com isso, ao receber um resultado considerado não satisfatório, um líder pode avaliar a equipe gerada e realizar as substituições necessárias para tratar possíveis pontos fracos encontrados.

5 TRABALHOS RELACIONADOS

Trabalho sobre formação de equipes. Diversos estudos na área de engenharia de software avaliam o uso de abordagens de otimização no processo de formação de equipes de desenvolvimento. Porém, os estudos se diferem quanto à estratégia utilizada para definir quais traços um profissional deve apresentar para agregar valor numa equipe. Alguns autores apontam que os profissionais mais capacitados tecnicamente são as melhores escolhas para integrar um time de sucesso [4]. Outros concluem que as preferências de um desenvolvedor quanto à realização de uma atividade também contribuem para o triunfo da equipe [3]. Por fim, observa-se também que outro fator relevante para o êxito de um grupo são as conexões sociais entre seus membros [1].

Para melhor entender a complexidade na atividade de seleção de membros para formação de um time, estudo na areá avaliou a utilização de *Agent-Based Modeling* em um modelo de dados composto por 116 projetos [5]. Dessa forma, cada time participante dos projetos foi avaliado, considerando a qualidade do produto entregue, o tempo e os recursos gastos. Os resultados sugerem que os times mais bem sucedidos foram os formados por profissionais com habilidades distintas e autossuficientes [5]. Tais resultados são estritamente relacionados ao objetivo da abordagem investigada neste trabalho. Entretanto, o presente artigo também utiliza as preferências de cada profissional na montagem dos times.

Seguindo os mesmos princípios, outro estudo utilizou o algoritmo *Artificial Bee Colony* [4] para definir qual indivíduo seria mais apto para assumir uma equipe a partir de suas *hard* e *soft skills*. Como método de avaliação, o time mais eficiente gerado via algoritmo foi comparado com uma equipe escolhida pelo gestor de projetos. Através de uma análise de risco, foi constatado que o grupo formado via algoritmo possuía uma taxa de risco maior, porém também era um time mais barato. Dessa forma, a pesquisa concluiu que é possível reduzir o número de falhas em um projeto, determinando a interdependência da habilidade de cada membro.

Numa abordagem diferente das anteriores, o estudo de Ahmad et al.[1] buscou formar equipes de desenvolvimento eficazes através da boa relação entre seus integrantes, realizando uma comparação de desempenho entre dois times. Para tanto, um dos times foi criado de forma tradicional, escolhido pelo líder do projeto, e outro via uma análise em redes sociais. O trabalho buscou identificar conexões entre os integrantes da equipe e gerar o grupo com maior sinergia entre seus participantes. Como resultado, foi possível observar que o time gerado via algoritmo alcançou resultados melhores quando comparado com o time escolhido pelo líder.

O estudo de Assavakamhaenghan et al. [2] utilizou de uma proposta distinta para a otimização do processo de formação de equipe de software, através da utilização de fatores técnicos associados a fatores sociais. Através de uma pesquisa qualitativa, buscou-se identificar, a partir do uso de técnicas de *machine learning*, um método para formar equipes eficientes através da habilidade técnica de seus integrantes e da sinergia entre seus componentes. O estudo aponta

que os resultados foram melhores do que os da *baseline* inicial, utilizando o método proposto.

Avaliação de habilidades técnicas. No intuito de tornar o processo de formação de equipes de desenvolvimento menos subjetivo, este trabalho busca avaliar as habilidades dos desenvolvedores através de dados. Outros estudos na área utilizam a mesma abordagem para identificar as habilidades técnicas de seus participantes. No principal deles [11], avaliam as características que descrevem um bom desenvolvedor, analisando suas competências técnicas e sociais. Nesse estudo, os autores utilizam dados coletados no Github para avaliar a capacidade técnica dos desenvolvedores, o número de *commits*, número de *issues*, número de comentários, número de projetos e número de linguagens utilizadas pelo desenvolvedor. O estudo atual reutiliza os mesmos artefatos para calcular as competências técnicas de seus participantes, utilizando tais características como entrada no processo de formação automática de times de desenvolvimento de software.

6 AMEAÇAS À VALIDADE

Primeiramente, ao considerar a habilidade e preferência de cada desenvolvedor como os fatores determinantes para o desempenho da equipe, o algoritmo avaliado pode ignorar outros aspectos sociais ou organizacionais capazes de influenciar os resultados apresentados por um time. Entretanto, observa-se que as respostas obtidas através dos líderes L1 e L2 apontam que critérios semelhantes são utilizados na prática. Além disso, observa-se o baixo número de opiniões coletadas durante a avaliação qualitativa dos líderes. Embora apenas duas entrevistas tenham sido realizadas, os líderes respondentes são profissionais com vários anos de experiência no mercado e familiaridade com os projetos e desenvolvedores da empresa escolhida. Por fim, um questionamento subsequente refere-se à escolha do algoritmo genético para a solução do problema apresentado. Dado o contexto e o universo de escolhas possíveis, a utilização de força bruta seria viável para considerarmos todas as combinações de desenvolvedores, selecionando a equipe com valores mais altos de habilidade e preferência. No entanto, caso a abordagem seja aplicada em um contexto onde o número de variáveis é maior, o poder computacional necessário para esse tipo de aplicação seria proibitivo. Dessa forma, considera-se satisfatória a escolha de combinações "quase ótimas", já que garante-se que ela pode ser realizada em um tempo computacional aceitável, independente do ambiente escolhido.

7 CONCLUSÃO

Neste trabalho investigou-se a aplicação de um método de otimização para montagem de equipes de trabalho, buscando identificar a viabilidade de utilização de uma metodologia quantitativa nesse processo. A avaliação se deu em uma empresa do mercado de desenvolvimento de software em Belo Horizonte. Para tanto, utilizou-se o algoritmo genético [3] para formação de equipes de desenvolvimento. Os dados de entrada relacionados às habilidades foram obtidos a partir da mineração de métricas disponíveis na plataforma de versionamento da empresa. Os dados relacionados às preferências, por sua vez, foram obtidas a partir de um questionário enviado aos desenvolvedores. O algoritmo foi executado para cada um dos projetos hipotéticos propostos, gerando como saída uma equipe considerada otimizada para a execução do projeto. Por fim, foi realizada uma avaliação comparativa entre as equipes geradas pelo algoritmo e as formadas pelos líderes.

Como resultado, observou-se que o algoritmo genético avaliado obteve sucesso na formação de equipes de desenvolvimento utilizando os membros da empresa. Em todos os seis projetos propostos, o algoritmo gerou equipes com um *fitness* de habilidade e preferência superior às equipes formadas por líderes experientes, conhecedores do contexto da empresa. Além disso, a avaliação dos líderes acerca das equipes formadas pelo algoritmo foram, em sua maioria, positivas. Dessa forma, podemos considerar como plausível a proposta da aplicação do algoritmos dentro de um cenário análogo ao de uma empresa real do mercado, representado pela empresa escolhida.

Trabalhos futuros podem integrar tanto *soft skills* quanto aspectos sociais e organizacionais dentro da função objetivo, a partir de uma análise das interações entre desenvolvedores e dos processos internos da empresa. Outro aspecto com grande potencial exploratório é a quantificação da habilidade dos desenvolvedores a partir de plataformas de versionamento de código. Elas permitem a mineração de representações mais precisas das habilidades dos profissionais, gerando resultados mais precisos por parte do algoritmo.

REFERENCES

[1] M. Ahmad, W. H. Butt, and A. Ahmad. 2019. Advance Recommendation System for the Formation of More Prolific and Dynamic Software Project Teams. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. 161–165.

[2] N. Assavakamhaenghan, P. Suwanworaboon, W. Tanaphantaruk, S. Tuarob, and M. Choetkiertikul. 2020. Towards Team Formation in Software Development: A Case Study of Moodle. In *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. 157–160.

[3] F d S Ferreira, JT Souza, and JS d V Silva. 2010. Formaçao de grupos de trabalho com algoritmo genético. In *III Congresso Tecnológico da InfoBrasil*. 1–5.

[4] Fitria and I Gusti Bagus Baskara Nugraha. 2018. Formation of Software Programmer Team Based on Skill Interdependency. In *2018 International Conference on Information Technology Systems and Innovation (ICITSI)*. 77–81.

[5] Shu-Chien Hsu, Kai-Wei Weng, Qingbin Cui, and William Rand. 2016. Understanding the complexity of project team member selection through agent-based modeling. *International Journal of Project Management* 34, 1 (2016), 82 – 93. http://www.sciencedirect.com/science/article/pii/S0263786315001507

[6] M. Aqeel Iqbal, F. A. Ammar, Adel Rashed Aldaihani, Tehmina Karamat Ullah Khan, and Asadullah Shah. 2019. Predicting Most Effective Software Development Teams by Mapping MBTI Personality Traits with Software Lifecycle Activities. In *2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS)*. 1–5. https://doi.org/10.1109/ICETAS48360.2019.9117370

[7] W. A. C. Prashandi and Abarnah Kirupananda. 2019. Automation of Team Formation in Software Development Projects in an Enterprise: What Needs to Improve?. In *2019 International Conference on Advanced Computing and Applications (ACOMP)*. 16–22. https://doi.org/10.1109/ACOMP.2019.00010

[8] Davide Spadini, Maurício Aniche, and Alberto Bacchelli. 2018. PyDriller: Python framework for mining software repositories. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*. ACM Press, New York, New York, USA, 908–911.

[9] Chong Wang, Zhong Luo, Luxin Lin, and Maya Daneva. 2017. How to Reduce Software Development Cost with Personnel Assignment Optimization: Exemplary Improvement on the Hungarian Algorithm. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. 270–279.

[10] Yuanyuan Zhang, Anthony Finkelstein, and Mark Harman. 2008. Search Based Requirements Optimisation: Existing Work and Challenges, Vol. 5025. 88–94.

[11] C. Zhou, S. K. Kuttal, and I. Ahmed. 2018. What Makes a Good Developer? An Empirical Study of Developers' Technical and Social Competencies. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 319–321.