Universidade Estadual de Campinas

Instituto de Computação

Exame de Qualificação Específico

Problemas de Coberturas Justas e Máximas

Candidata: Ana Paula dos Santos Dantas

Orientador: Prof. Dr. Zanoni Dias

Coorientador: Prof. Dr. Cid Carvalho de Souza

Resumo

Neste projeto, abordamos problemas de cobertura considerando conceitos de justiça. Dados um conjunto universo $\mathcal{U}$, uma família de subconjuntos $\mathcal{S}$, uma função C de coloração dos elementos de $\mathcal{U}$ e um inteiro k, dizemos que uma cobertura $X \subset \mathcal{S}$ de tamanho k é justa quando a quantidade de elementos cobertos é distribuída igualmente entre todas as classes de cores. Dada uma função de peso para os elementos de $\mathcal{U}$, definimos o problema de Cobertura Justa Máxima (FMC), cujo objetivo é encontrar uma cobertura de tamanho k que seja justa e que a soma dos pesos dos elementos cobertos seja máxima. Esse problema foi proposto como uma das possíveis abordagens de evitar casos de injustiça algorítmica, em que parte da população pode ser prejudicada quando visamos apenas a maximização dos ganhos. Estudamos o FMC, e algumas variações desse problema, de modo a desenvolver soluções eficientes que possam ser usadas em aplicações como Integração de Dados e Localização de Instalações. Para tal, propomos a abordagem desses problemas em três linhas: análise da complexidade computacional; desenvolvimento de algoritmos exatos, aproximados e heurísticas para a resolução de instâncias; e execução de experimentos computacionais com os algoritmos desenvolvidos.

Palavras-chave: *Cobertura Justa Máxima; Justiça Algorítmica; Otimização Combinatória; Pesquisa Operacional.*

Abstract

In this project, we tackle covering problems considering concepts of fairness. Given a universe set $\mathcal{U}$, a family of subsets $\mathcal{S}$, a coloring function C of the elements from $\mathcal{U}$, and an integer k, we say that a cover $X \subset \mathcal{S}$ of size k is fair if the number of covered elements is equally distributed between all the color classes. Given a weight function for the elements of $\mathcal{U}$, we define the problem of Fair Maximum Covering (FMC), whose objective is to find a cover of size k that is fair and whose sum of the weights of the covered elements is maximum. This problem was proposed in an attempt to avoid cases of algorithmic injustice, where part of the population may be harmed when we consider only the maximization of profits. We study the FMC, and a few variations of the problem, to develop efficient solutions that can be used in applications such as Data Integration and Facility Location. For such, we propose to approach these problems in three lines: analysis of computational complexity; development of exact, approximation, and heuristic algorithms; and the execution of computational experiments with the proposed algorithms.

Keywords: *Fair Maximum Cover; Algorithmic Fairness; Combinatorial Optimization; Operations Research.*

1 Introdução

O uso de algoritmos como ferramenta para a tomada de decisões vem crescendo consideravelmente nas últimas décadas. Eles são usados em análises nos mais diversos campos, desde a concessão de liberdade condicional [8] e predição de emergências em unidades de tratamento intensivo [6] até a seleção de alunos em universidades [31]. Vale notar que essas ferramentas são usadas não só em situações onde já esperamos encontrar algoritmos, como ferramentas de busca na *web*, mas também em diversos aspectos do nosso cotidiano e que por muitas vezes não percebemos o seu uso, como publicidade direcionada. Tais inovações trazem benefícios, como rapidez nas decisões judiciais e nas respostas em situações de emergências, assim como potenciais melhoras no desempenho geral de alunos.

Cathy O'Neil mostra em seu livro "Algoritmos de Destruição em Massa" [23] diversas situações nas quais algoritmos foram empregados explicitamente na tomada de decisões. Dentre elas, análise e concessão de crédito em bancos, propagandas direcionadas e avaliação do desempenho de professores da rede pública. No livro, O'Neil enfatiza o potencial que essas ferramentas têm para impactar a sociedade e como a população mais vulnerável acaba sendo a mais prejudicada pelos deslizes e má utilização dessas ferramentas.

As ferramentas não estão isentas das aflições que assolam nossa sociedade, como discriminações raciais e de gênero. Por exemplo, um estudo feito em 2016 identificou que o sistema usado para atribuir riscos de reincidência criminal no estado da Florida (EUA) era duas vezes mais provável de indicar um réu negro com alto risco. De modo que esse valor alto para o risco de reincidência implicava em sentenças mais duras e menores chances de liberdade condicional para réus negros [2].

Outro estudo, também de 2016, analisou a cobertura de serviços de entrega no mesmo dia oferecidos por uma plataforma *online*. Com essa análise, percebeu-se que a grande maioria dos bairros não atendidos tinha uma população predominante negra ou pertencente a outras etnias minoritárias no território americano [14].

Casos de injustiça algorítmica, mais especificamente de racismo algorítmico, acontecem há bastante tempo, com os primeiros casos sendo reportados em meados de 2010. Numa reportagem da revista *Time*, por exemplo, é relatado um caso onde o *software* de uma câmera fotográfica tem dificuldade em identificar olhos asiáticos, indicando-os como fechados em fotografias [26]. Na mesma reportagem é citado também um caso em que outro *software* de câmera falhou em identificar e seguir o rosto de um homem negro. Recentemente, um vídeo com imagens de homens negros foi rotulado em uma rede social como um vídeo sobre primatas [21]. Silva [27]

apresenta um compilado de notícias cujo foco são casos de racismo algorítmico, onde é notável que esses acontecimentos têm se tornado mais comuns nos últimos anos. Tais reportagens demonstram as formas como o racismo algorítmico vem sendo identificado.

Desde a divulgação desses casos, a comunidade vem reagindo para evitar que isso se repita. Grande parte dessa reação tem considerado como tratar os dados de entrada dos algoritmos e como estabelecer critérios de quantificação do nível de justiça. A ideia de que os algoritmos devem continuar ignorantes a características potencialmente descriminatórias como raça e gênero foi contestada no trabalho de Kleinberg *et al.* [16], onde foram apresentados experimentos com modelos de predição de sucesso de alunos, usados no processo de admissão em universidades. Nesses experimentos, foi demonstrado que o algoritmo que considera as caraterísticas de raça teve melhor desempenho comparado ao caso onde a raça dos aplicantes era desconsiderada e ao caso onde foram feitos pré-processamentos na base de dados para remover características que são correlacionadas com raça.

Na mesma linha, Lin *et al.* [20] apresentaram métodos para encontrar regiões de uma base de dados com pouca cobertura, isto é, combinações de classes de atributos com poucas amostras. Os autores realizaram experimentos com algoritmos de classificação, para os quais a entrada é um conjunto de dados rotulados, usados na fase de treinamento. O objetivo desses algoritmos é determinar o rótulo de amostras que previamente não possuíam rótulos [1]. Nesses experimentos, os autores favorecem a inserção de amostras das áreas com pouca cobertura na fase de treinamento, o que resultou em melhorias na acurácia de todas as três bases de dados testadas.

Os dois trabalhos mencionados [16, 20], concluem que, para deixarmos os algoritmos mais justos, precisamos incluir deliberadamente amostras de todas as classes. Considerando isso, Asudeh *et al.* [3] definiram um problema de otimização para selecionar amostras de maneira justa. Nesse problema, uma amostra é um conjunto de elementos coloridos de acordo com os valores das classes de cada atributo. Pesos são atribuídos a cada elemento e o objetivo é encontrar uma seleção de amostras cujos pesos tenham soma máxima e cada classe de atributo apareça no mesmo número de amostras. Esse problema foi chamado de Cobertura Justa Máxima (*Fair Maximum Coverage* – FMC).

Asudeh *et al.* [3] também notaram que esse problema de cobertura pode ser aplicado em problemas de Localização de Facilidades, sendo necessário, nesse caso, escolher instalações que atendam o mesmo número de clientes de classes distintas. Essa aplicação poderia evitar os casos apresentados por Ingold *et al.* [14] e Guse [12], onde minorias tiveram acesso desproporcional aos serviços de entrega e meios de transporte alternativo. Podemos também expandir a definição do FMC para diversos outros conceitos de justiça e continuar tratando-o como um problema de

cobertura. Por exemplo, podemos restringir que o número de elementos cobertos de uma dada classe seja proporcional à sua frequência no conjunto universo.

O FMC é um problema NP-difícil [3] e possui algoritmos com fator de aproximação $(1 - 1/f)^f$, onde f é a frequência máxima de um elemento do conjunto universo nos subconjuntos. Por se tratar de um problema recente, a literatura específica ainda é pequena e apresenta algumas lacunas.

O restante dessa proposta está organizado da seguinte forma. Na Seção 2, denotamos as convenções de notação e definições básicas usadas no texto. Na Seção 3, apresentamos um breve resumo da literatura dos problemas relacionados, como problemas de coberturas e justiça algorítmica. Na Seção 4, listamos os objetivos gerais e específicos. Na Seção 5, detalhamos a metodologia empregada no desenvolvimento deste projeto. Na Seção 6, listamos as tarefas a serem realizadas e um cronograma para a execução das mesmas. Na Seção 7, apresentamos formulações para diferentes versões dos problemas de cobertura abordados.

2 Notações e Definições

Nesta seção apresentamos algumas das definições e notações usadas ao longo deste projeto. Definimos como $u_1, u_2, \ldots, u_m$ os elementos de um conjunto universo $\mathcal{U}$. Chamamos de ***família de subconjuntos*** (ou apenas *família*) um conjunto $\mathcal{S}$ que é formado por subconjuntos $\mathcal{S}_i$ do conjunto universo $\mathcal{U}$. Uma ***cobertura*** em sua interpretação clássica é definida como uma subfamília $\mathcal{Z} \subseteq \mathcal{S}$ tal que a união de todos os subconjuntos pertencentes a essa subfamília $\mathcal{Z}$ é igual ao conjunto universo, ou seja, $\bigcup_{\mathcal{S}_i \in \mathcal{Z}} \mathcal{S}_i = \mathcal{U}$.

Dizemos que um elemento u_j é coberto por $\mathcal{Z}$, se existe pelo menos um subconjunto $\mathcal{S}_i$ na subfamília ao qual u_j pertence, isto é, existe $\mathcal{S}_i \in \mathcal{Z}$ tal que $u_j \in \mathcal{S}_i$. Podemos dizer que em uma cobertura todos os elementos $u_j \in \mathcal{U}$ devem ser cobertos. Uma **k-*cobertura*** X é uma subfamília de $\mathcal{S}$ que contém exatamente k subconjuntos. Diferentemente de uma cobertura, em uma k-cobertura não é necessário que todos os elementos u_j sejam cobertos. Dada uma família $\mathcal{F}$, definimos a frequência de um elemento u_j como o número de subconjuntos $\mathcal{S}_i$ aos quais o u_j pertence. Definimos também f como a maior das frequências f_j e a como a cardinalidade máxima de um conjunto $\mathcal{S}_i$.

Definimos $\mathcal{C}$ como um conjunto de tamanho χ formado por cores distintas (representadas por inteiros positivos), isto é, $\mathcal{C} = \{1, 2, \ldots, \chi\}$. Chamamos de ***coloração*** uma função $C : \mathcal{U} \to \mathcal{C}$ que define uma cor para cada elemento de $\mathcal{U}$. Definimos como C_c o conjunto formado por todos os elementos coloridos com a cor c, tal que $C_c = \{u_j \mid C(u_j) = c\}$. Definimos q_c como a cardinalidade

de C_c, isto é, o número de elementos coloridos com a cor c. Seja p_i o número de elementos da cor c_i cobertos pela k-cobertura X. Dizemos que uma k-cobertura X é ***justa***, se X cobre o mesmo número de elementos de cada cor, ou seja, $p_1 = p_2$ para todo $c_1, c_2 \in \mathcal{C}$.

Seja w uma função de pesos para os elementos de $\mathcal{U}$, tal que $w : \mathcal{U} \to \mathbb{R}$. Dizemos que uma k-cobertura X é **máxima** se a soma dos pesos dos elementos $u_j \in X$ é a maior possível. Podemos agora formalizar uma definição para o problema de Cobertura Justa Máxima (*Fair Maximum Coverage* – FMC), apresentado a seguir na Definição 2.1.

Definição 2.1. Cobertura Justa Máxima – FMC

Entrada: Conjunto universo $\mathcal{U}$, função de coloração C, conjunto de subconjuntos $\mathcal{S}$, função de pesos w para os elementos de $\mathcal{U}$ e um inteiro positivo k.

Objetivo: Encontrar uma k-cobertura justa, tal que a soma dos pesos dos elementos cobertos é máxima.

Na Figura 1 ilustramos os conceitos definidos. Nesse exemplo, temos um conjunto universo $\mathcal{U}$ com 12 elementos $\{u_1, u_2, \ldots, u_{12}\}$. Os elementos de $\mathcal{U}$ foram coloridos com três cores ($\chi = 3$): ●, ● e ●. À direita, temos a família de conjuntos de $\mathcal{U}$. Com este exemplo, podemos criar uma cobertura selecionando os subconjuntos $\mathcal{S}_1$, $\mathcal{S}_3$, $\mathcal{S}_4$, $\mathcal{S}_5$, de modo que todos os elementos são cobertos. Também podemos obter uma k-cobertura para qualquer $k \in \{1, 2, 3, 4, 5\}$.

Supondo que $k = 2$ e que o peso dos elementos u_j da Figura 1 é igual a um, isto é, $w(u_j) = 1, \forall u_j \in \mathcal{U}$, uma possível 2-cobertura é $X = \{\mathcal{S}_1, \mathcal{S}_5\}$, que cobre os elementos u_1, u_2, u_3, u_4, u_6, u_7, u_8, u_{10}, u_{11} e u_{12}. Essa 2-cobertura é máxima mas não é justa, pois o número de elementos cobertos com a cor ● é maior que o número de elementos cobertos com as cores ● e ●. Contudo, temos uma 2-cobertura justa com $Z = \{\mathcal{S}_1, \mathcal{S}_4\}$, que cobre os elementos u_1, u_3, u_4, u_6, u_7, u_8, u_9, u_{10} e u_{12}, onde são cobertos três elementos de cada cor e soma dos pesos é igual a nove. Note que considerando o conceito de justiça e o peso uniforme, o total da soma dos pesos é limitado pelo tamanho da cor menos frequente em $C(\mathcal{U}) \times \chi$.

Existe um caso particular do FMC onde podemos usar grafos para representar uma instância. Nesse caso, restringimos a frequência f_j de cada elemento u_j de $\mathcal{U}$ a exatamente dois subconjuntos $\mathcal{S}_i$. Assim, criamos um grafo cujas arestas representam os elementos do conjunto universo e os vértices representam subconjuntos dos elementos. Note que, o grafo criado respeita a restrição de frequência pois uma aresta incide em exatamente dois vértices. Dizemos que os vértices cobrem as arestas. A versão do problema de Cobertura Justa e Máxima em grafos é chamada de NODE-FMC (do inglês *Fair Maximum Node Covering*).

Na Figura 2, ilustramos a equivalência entre um grafo e uma família de subconjuntos cuja frequência f_j é dois para todo elemento u_j. Note que usamos cores nos vértices à esquerda apenas para mapear os subconjuntos. Nesse exemplo, nomeamos todas as arestas do grafo que representam os elementos do conjunto universo. Considerando a aresta u_2 no grafo, ela conecta os vértices v_2 e v_5, coloridos com as cores verde e laranja, respectivamente. Logo, o elemento u_2 pertence a ambos os conjuntos sombreados de verde e laranja na segunda parte da figura. Vale notar que o grau de um vértice é a cardinalidade do subconjunto da instância.

Asudeh *et al.* [3] apresentaram algumas aplicações para os problemas de coberturas justa e máximas. Dentre elas, duas merecem destaque: Localização de Facilidades e Integração de Dados. Na aplicação de Localização de Facilidades estamos interessados em selecionar facilidades que atendam aos clientes de modo maximizar os lucros, mas que também sejam justas, visando evitar casos como o citado por Ingold e Soper [14], onde os bairros cuja população era predominantemente negra não eram atendidos pelo serviço de entrega no mesmo dia. Já em Integração de Dados, estamos interessados em selecionar subconjuntos de dados que respeitem o conceito de justiça, de modo a mitigar problemas de viés nas aplicações de tais bases de dados.

A aplicação em Localização de Facilidades possui uma tradução mais direta para os problemas de k-cobertura justa, haja visto que existem modelagens do problema com coberturas de conjuntos. Desse modo, adicionamos a informação sobre a classe a qual o elemento pertence para obtermos uma instância para os problemas de coberturas justas. Já para a aplicação em Integração de Dados é necessário criar um protocolo para transformar uma base de dados em uma instância de cobertura. A seguir apresentamos uma forma de transformar uma base de dados em formato tabular em uma instância do problema de cobertura.

No formato tabular de uma base de dados, chamamos uma linha de *amostra* e uma coluna de *atributo*. Uma amostra representa um indivíduo da população e um atributo representa uma

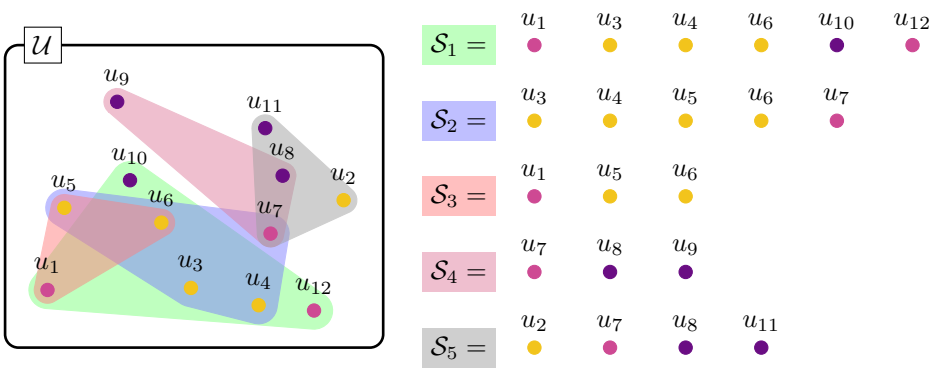


Figura 1: Exemplo de conjunto universo $\mathcal{U}$ (esquerda) e família de subconjuntos $\mathcal{S}$ (direita). Os elementos de $\mathcal{U}$ são coloridos com três cores e os elementos de $\mathcal{S}$ são subconjuntos de $\mathcal{U}$.

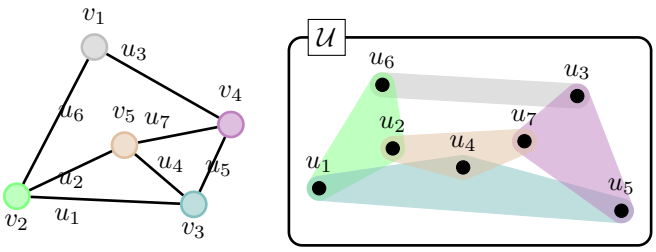


Figura 2: Transformando um grafo em conjunto universo e família de subconjuntos.

id amostra	atributo 1	atributo 2
1	**alto**	**largo**
2	**baixo**	**largo**
3	**baixo**	**fino**
4	**baixo**	**fino**
5	**baixo**	**fino**
6	**médio**	**fino**
7	**médio**	**largo**
8	**alto**	**fino**
9	**alto**	**fino**
10	**baixo**	**fino**
11	**alto**	**largo**
12	**baixo**	**fino**

Tabela 1: Exemplo de uma base de dados em formato tabular com 12 amostras e 2 atributos categóricos.

característica dos indivíduos. Atributos podem ter diferentes tipos, contudo consideramos apenas atributos discretos (ou categóricos). Nessa aplicação o objetivo é selecionar amostras de modo que a cobertura dos diferentes tipos de atributos sejam representados igualmente na k-cobertura, na versão mais básica do problema de k-cobertura justa. Assim, uma amostra deve ser interpretada como um dos subconjuntos $\mathcal{S}_\ell$ de $\mathcal{S}$, ou um vértice na versão em grafos.

No exemplo ilustrado na Tabela 1, o primeiro atributo tem 3 classes **alto**, **médio** e **baixo** e o segundo atributo tem 2 classes **largo** e **fino**. Com uma tradução mais direta interpretamos cada célula da Tabela 1 como um elemento do conjunto universo. A coloração é feita diretamente com base no tipo do atributo. Dessa forma, temos uma versão mais peculiar onde cada elemento tem frequência unitária, isto é, pertence a exatamente um subconjunto.

Apresentamos uma interpretação gráfica dessa transformação na Figura 3 considerando apenas as três primeiras amostras. A ideia é criar um vértice falso para cada classe dos atributos considerados, com a restrição de que esses vértices falsos não podem ser utilizados na cobertura. Assim, cada amostra está conectada aos vértices falsos que correspondem aos valores que a compõem. Por exemplo, o valor do atributo 1 da amostra 1 é **alto**. Logo essa amostra está conectada ao vértice falso que representa **alto**. Para simplificar ainda mais a visualização,

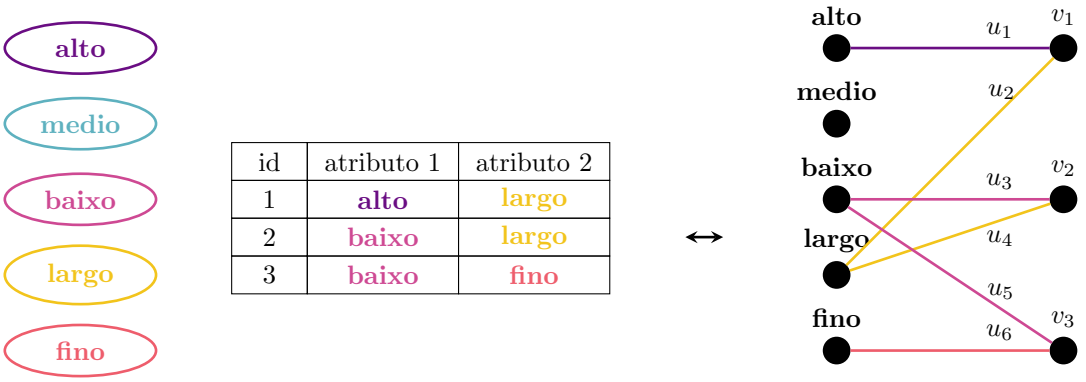


Figura 3: Transformando uma tabela em um grafo que pode ser usado como entrada para o problema.

transformamos uma amostra em um vértice válido para a cobertura, como mostra a segunda parte da Figura 3.

Vale notar que é necessário tomar um cuidado especial na interpretação do conceito de justiça quando em aplicações de Integração de Dados. Nessa aplicação, é necessário considerar os conceitos de justiça para cada atributo da base de dados de forma individual. Isto é, a quantidade de elementos cobertos em uma dada classe c só deve ser igual à quantidade de elementos de uma classe d, se c e d são classes de um mesmo atributo. Por exemplo na base de dados da Tabela 1, o número de elementos de cor amarelo não precisa ser igual ao número de elementos cobertos de cor rosa.

3 Revisão Bibliográfica

No problema clássico de Cobertura por Conjuntos (*Set Covering Problem* – SCP), temos um conjunto universo $\mathcal{U}$ e uma família S de subconjuntos como entrada. O objetivo do SCP é descobrir se existe uma subfamília S^* que cobre todos os elementos do conjunto universo, isto é, $\bigcup_{S_i \in S^*} S_i = \mathcal{U}$. O SCP é conhecidamente NP-completo [15]. A versão de otimização, onde a soma dos pesos dos elementos cobertos deve ser minimizada, não pode ser aproximado com um fator logarítmico, a menos que $P = NP$ [24]. O SCP possui aplicações em áreas como alocação de turnos [4], posicionamento de sensores [28, 17], posicionamento de postos de serviços emergenciais [5], entre outras.

O SCP é um problema bastante estudado na literatura. As abordagens mais recentes para tratar esse problema usam meta-heurísticas bio-inspiradas como Colônia de Formigas (*Ant Colony Optimization*) [25], Otimização por Enxame de Partículas (*Particle Swarm Optmization*) [17] e o Algoritmo de Otimização baseado em nuvens de Gafanhotos (*Grasshopper Optmization*

Algorithm) [30].

Outro resultado interessante é uma formulação matemática alternativa, que tem como proposta reduzir o tratamento da inviabilidade das instâncias e das redundâncias nos conjuntos [7]. Nessa formulação, a função objetivo do modelo maximiza a diferença entre o custo de selecionar a subfamília e o ganho ao cobrir um determinado elemento, enquanto que o modelo clássico minimiza os custos de selecionar a subfamília. Lanza *et al.* [18] apresentaram estudos com bons resultados utilizando essa formulação para guiar procedimentos de busca em meta-heurísticas.

O SCP também é apresentado na literatura como um problema de minimização de custos [9]. Deste modo, é necessário que uma função de custo atribua um valor para cada conjunto presente na família S e o objetivo passa a ser encontrar uma subfamília que cubra todos os elementos de $\mathcal{U}$ que tenha custo mínimo.

Ao considerarmos as aplicações do SCP, algumas restrições podem surgir, como limites no custo total dos elementos cobertos ou no número de subfamílias da cobertura. Quando essa última limitação é considerada na literatura, uma cobertura deve ser modificada para permitir que alguns elementos de $\mathcal{U}$ não sejam cobertos e o objetivo passa a ser maximizar os pesos dos elementos cobertos. Esse problema é conhecido como Cobertura Máxima por k-Conjuntos (*Maximum k-Set Cover* – MKSC). Utilizando pesos unitários, o objetivo do MKSC pode ser interpretado como uma maximização dos elementos cobertos.

O MKSC é um problema NP-difícil [15]. Um dos algoritmos mais conhecidos para o MKSC é um algoritmo guloso, que é também uma H_k-aproximação, com $H_k = \sum_{i=1}^{k} \frac{1}{i}$, que foi proposto por Chvatal [9]. Esse fator de aproximação foi posteriormente melhorado para $H_k - \frac{k-1}{8k^9}$ para funções de peso arbitrárias [13] e $H_k - \frac{196}{390}$ para a versão sem pesos [19].

4 Objetivos

O objetivo principal deste trabalho é desenvolver ferramentas eficientes que possam ser usadas para mitigar problemas de injustiça algorítmica. Para isso, abordaremos os problemas de Cobertura Justa e Máxima (FMC); Cobertura Justa e Proporcional Máxima (FPMC); Cobertura Quase Justa Máxima (QFMC); Cobertura Quase Justa e Proporcional Máxima (QFPMC); e suas respectivas versões em grafos: Cobertura Justa Máxima por Vértices (NODE-FMC); Cobertura Justa e Proporcional Máxima por Vértices (NODE-FPMC) Cobertura Quase Justa Máxima por Vértices (NODE-QFMC); e Cobertura Quase Justa e Proporcional Máxima por Vértices (NODE-QFPMC).

Abordaremos esses problemas sob as perspectivas teórica e experimental. Durante a

abordagem teórica, estudaremos a complexidade dos problemas apresentados na Seção 2, bem como das versões que podem surgir durante o desenvolvimento deste projeto. Além disso, também serão desenvolvidos algoritmos exatos, heurísticas e algoritmos de aproximação para os problemas tratados.

Desenvolveremos experimentos computacionais com os algoritmos criados, comparando o desempenho em tempo de execução e qualidade da solução obtida por cada abordagem. Não existem instâncias propostas na literatura para os problemas tratados neste projeto. Utilizaremos instâncias geradas a partir de bases de dados das aplicações, bem como instâncias artificiais, com as quais poderemos controlar melhor as características. Todos os experimentos serão validados com métodos estatísticos. Também serão criados conjuntos de instâncias artificiais e baseadas nas aplicações para serem usados nos experimentos.

5 Metodologia

Nesta seção, detalhamos como este projeto será desenvolvido expandindo os objetivos apresentados na Seção 4.

A primeira parte deste projeto tem como foco a obtenção de soluções exatas para o FMC e demais problemas. Inicialmente, estudamos um modelo de Programação Linear Inteira (PLI) para o FMC. O modelo apresentado por Asudeh *et al.* [3] é uma extensão de uma formulação básica para o problema de Cobertura por Conjuntos. Esse modelo será apresentado e discutido com mais detalhes na Seção 7.

Asudeh *et al.* [3] usaram uma versão desse modelo para obter limitantes dos problemas FMC, NODE-FMC, SEG-FMCe Δ-BAL-FMC, onde foram aplicadas algumas relaxações e adicionadas desigualdades válidas. Neste projeto, utilizaremos esse modelo como base para obtenção de soluções ótimas para o FMC e NODE-FMC. Em seguida, esse modelo será adaptado para as demais versões dos problemas de cobertura definidos neste projeto. Além disso, também estudaremos técnicas para a melhoria do modelo.

Algumas versões do problema, que surgem ao relaxarmos restrições ou modificarmos o critério de justiça, não possuem uma classificação de complexidade algorítmica conhecida. De modo que estudaremos essas versões a fim de classificá-las, para uma subsequente escolha de métodos de solução. Algoritmos de aproximação produzem, em tempo polinomial, soluções cujo valor da função objetivo estão a uma distância garantida do valor ótimo [29]. Esses algoritmos são usados para tratar problemas NP-difíceis. Essas duas tarefas envolvem estudos de teoremas e provas presentes na literatura que servirão de base para a classificação dos problemas

e desenvolvimento de algoritmos.

Também trataremos os problemas de cobertura justa com heurísticas. Inicialmente, usaremos a meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP) [11]. A essência dessa meta-heurística consiste na execução de duas fases básicas: a criação de uma solução e a busca local. Essas duas fases são repetidas até que um critério de parada, geralmente o número de iterações, seja atendido.

Na primeira fase do GRASP, uma solução é construída iterativamente com base em uma Lista Restrita de Candidatos (*Restricted Cadidate List* – RCL). Essa lista contém uma porcentagem α das próximas escolhas na construção de uma solução, ordenados de maneira gulosa. A cada iteração da construção, uma RCL é construída e um elemento pertencente a ela é selecionado aleatoriamente para fazer parte da solução. Já na segunda fase do GRASP, é realizada uma busca na vizinhança da solução encontrada, a fim de encontrar um ótimo local. Note que ao final de cada iteração do GRASP, obtemos uma solução viável, de modo que apenas a melhor é retornada.

Escolhemos trabalhar inicialmente com esta meta-heurística, por sua robustez e simplicidade, possuindo poucos parâmetros para ajuste. Além disso, as estruturas dessa servem de base para o uso de outras meta-heurísticas. Por exemplo, ao definirmos vizinhanças para a busca local na segunda fase do GRASP, podemos utilizá-las também para criar algoritmos baseados na meta-heurística *Variable Neighborhood Search* (VNS) [22], que explora as vizinhanças de uma solução, passando de uma vizinhança para outra quando não há melhora na solução.

Realizaremos experimentos computacionais com os modelos matemáticos, com as heurísticas e com os algoritmos de aproximação. Para isso, criaremos instâncias artificiais e instâncias baseadas nas áreas de aplicação do problema, isto é, Integração de Dados (*Data Integration*) e Localização de Facilidades (*Facility Location*). Para as instâncias baseadas na aplicação de Integração de Dados, podemos usar bases de dados para classificação. Já para as instâncias baseadas na aplicação de Localização de Facilidades, podemos usar, por exemplo, dados de censo demográfico para coloração dos elementos do conjunto universo e criação da família de subconjuntos. Essas instâncias serão usadas como *benchmark* para a análise de todas as abordagens propostas neste projeto. Como parte dos experimentos, faremos comparações a fim de atestar a qualidade das diferentes abordagens. Essas comparações serão validadas utilizando métodos estatísticos, sempre que aplicável.

6 Cronograma

Nesta seção, apresentamos uma lista das etapas a serem desenvolvidas durante a vigência desse projeto. Na Figura 4 apesentamos a distribuição destas tarefas no período de execução.

1. Obtenção dos créditos obrigatórios em disciplinas do Programa de Pós-Graduação.
2. Participação no Programa de Estágio Didático (PED).
3. Revisão bibliográfica.
4. Escrita do projeto e experimentos iniciais com o modelo de programação linear inteira da literatura.
5. Obtenção de soluções exatas através de modelos matemáticos para o FMC, FPMC e QFMC, e suas respectivas versões em grafos.
6. Análise de bases de dados reais para as aplicações em integração de dados e localização de instalações.
7. Documentação dos resultados.
8. Análise da complexidade algorítmica dos problemas FPMC e QFMC.
9. Exame de Qualificação Específico (EQE).
10. Estudo e desenvolvimento de algoritmos de aproximação para o FMC, FPMC e QFMC, e suas respectivas versões em grafos.
11. Desenvolvimento de abordagens heurísticas para o FMC, FPMC e QFMC, e suas respectivas versões em grafos.
12. Escrita do documento da tese.
13. Defesa da tese de doutorado.

Vale ressaltar que o cronograma apresentado nesta seção pode ser alterado, uma vez que alguns resultados obtidos podem ser mais promissores que outros, o que nos faria dedicar mais tempo em algumas etapas em detrimento de outras.

7 Modelando os Problemas de Coberturas Justas

Nas seções a seguir apresentamos algumas modelagens possíveis para diferentes interpretações dos problemas de coberturas justas. Analisamos dois conceitos de justiça, bem como relaxações para os problemas.

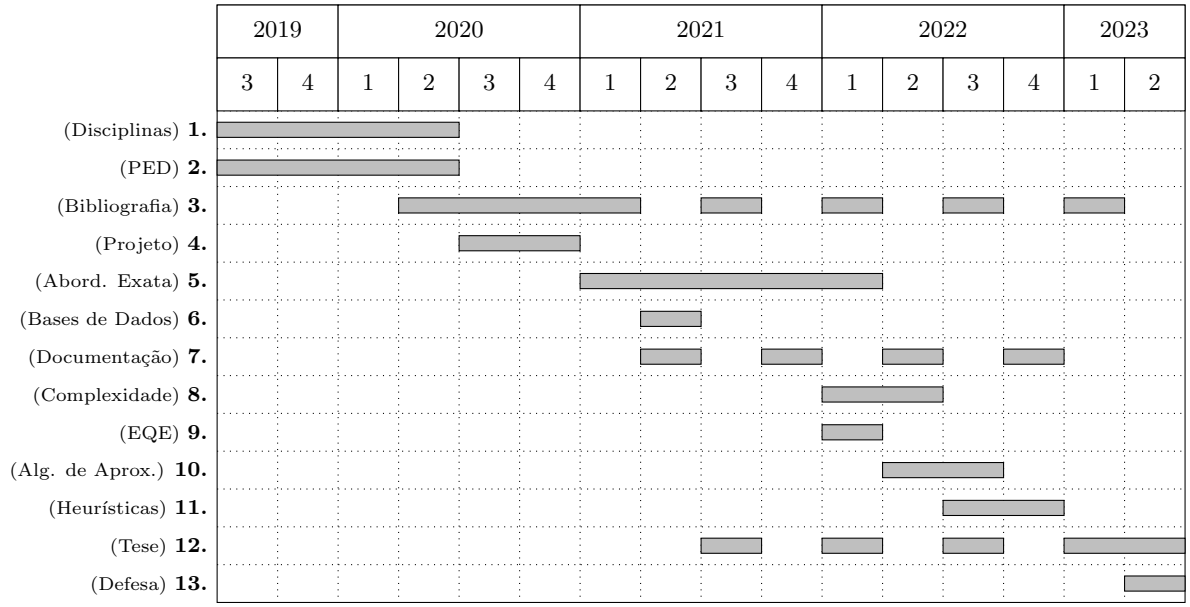


Figura 4: Cronograma de atividades divididas em trimestres.

7.1 Justiça como Igualdade

O problema de Cobertura Justa Máxima (*Fair Maximum Covering* - FMC) foi inicialmente definido por Asudeh *et al.* [3] considerando um conceito de justiça onde todas as classes devem ter a mesma representação em uma solução ótima. Isto é, o número de elementos cobertos pertencentes a uma classe deve ser igual ao número de elementos cobertos pertencentes às demais classes. O objetivo do FMC é encontrar uma cobertura justa cuja soma dos pesos dos elementos cobertos é máxima e que respeite o orçamento definido na entrada, esse orçamento define o número de subconjuntos na cobertura. A Definição 2.1, formaliza essa versão do problema.

Contudo essa versão do problema de justiça em coberturas é bastante restritiva, tanto no seu conceito de justiça quanto no orçamento, onde devemos selecionar exatamente k subconjuntos. Apresentamos a seguir o modelo $\mathcal{F}$ de Programação Linear Inteira para o FMC. Nesse modelo são usadas duas variáveis de decisão: $x_j \in \{0,1\}$, que indica se um elemento u_j é coberto pela solução, e $y_\ell \in \{0,1\}$ indica se um subconjunto $\mathcal{S}_\ell$ pertence à cobertura. Nesse modelo também usamos m para indicar o número de elementos do conjunto universo, n para indicar o número de subconjuntos. Representamos o orçamento com a constante k, os pesos dos elementos u_j com as contantes $w(u_j)$ e o conjunto de cores usadas na classificação dos elementos u_j com a constante $\mathcal{C}$. Por último, dividimos os elementos cobertos com uma cor c qualquer em grupos representados

por C_c.

$$\mathcal{F}: \quad \max \quad \sum_{j=1}^{m} w(u_j)x_j \tag{1a}$$

$$\text{s. a.} \qquad x_j \leq \sum_{u_j \in \mathcal{S}_\ell} y_\ell \quad \forall\, j \in \{1,2,\ldots,m\} \tag{1b}$$

$$y_\ell \leq x_j \qquad \forall\, \ell \in \{1,2,\ldots,n\}, \forall\, j \in \{1,2,\ldots,m \mid u_j \in \mathcal{S}_\ell\} \tag{1c}$$

$$\sum_{\ell=1}^{n} y_\ell = k \tag{1d}$$

$$\sum_{u_j \in C_c} x_j = \sum_{u_i \in C_d} x_i \quad \forall\, c,d \in \mathcal{C},\ c < d \tag{1e}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in \{1,2,\ldots,m\} \tag{1f}$$

$$y_\ell \in \{0,1\} \qquad \forall\, \ell \in \{1,2,\ldots,n\} \tag{1g}$$

A Equação (1a) representa a função objetivo, onde a soma dos pesos dos elementos cobertos é maximizada. Em seguida, temos duas Restrições (1b) e (1c), que são adicionadas para modelar a cobertura. Na primeira, Restrição (1b), dizemos que para um elemento u_j ser coberto, é necessário que pelo menos um dos subconjuntos $\mathcal{S}_\ell$ aos quais ele pertence esteja na cobertura. Já na segunda, Restrição (1c), impõe que se um subconjunto $\mathcal{S}_\ell$ está na cobertura, então todos os elementos que o compõem devem ser cobertos. A Restrição (1d) restringe o número de subconjuntos na solução, que deve ser igual ao orçamento estipulado pela instância. A Restrição (1e) é responsável por assegurar a característica de justiça a uma solução. Por último, as Restrições (1f) e (1g) definem as variáveis de decisão como binárias.

7.1.1 Aplicando Limitações nas Instâncias

Asudeh *et al.* [3] definiram algumas versões que aplicam restrições nas instâncias ou definem o problema sob uma ótica diferente. A seguir, apresentamos esses problemas.

O problema de Cobertura Justa Máxima Δ-Balanceada (Δ-*Balanced Fair Maximum Coverage* – Δ-BAL-FMC) restringe que o número de elementos de cada cor em um conjunto $\mathcal{S}_i$ esteja a uma distância no máximo Δ do valor $\frac{|\mathcal{S}_i|}{\chi}$. Isto implica que cada conjunto $\mathcal{S}_i$ é quase balanceado.

Outro problema que altera a instância é o problema de Cobertura Justa Máxima com Segregação (*Segregated Fair Maximum Coverage* – SEG-FMC). Nesse problema, cada subconjunto $\mathcal{S}_i$ contém apenas elementos de uma mesma cor. Em ambos os problemas, Δ-BAL-FMC e SEG-

FMC, o objetivo é encontrar uma cobertura justa que maximize a soma dos pesos dos elementos cobertos.

7.1.2 Relaxando Restrições na Formulação do Problema

No modelo $\mathcal{F}$, que foi apresentado por Asudeh *et al.* [3], as Restrições (1d) e (1e) definem as características principais do FMC: orçamento fixo e justiça através da igualdade nos números de elementos cobertos. Essas também são as restrições que juntas reduzem bastante o espaço de soluções e, por vezes, tornam instâncias inviáveis. Ao analisarmos o problema e suas aplicações podemos modificar a interpretação de alguns pontos, em especial sobre essas duas restrições. A depender da aplicação, podemos permitir que o conceito de justiça seja violado em no máximo τ unidades. Isto é, em uma aplicação na qual é imprescindível que o orçamento seja atendido, permitimos soluções que tenham uma diferença de no máximo τ unidades no número de elementos cobertos entre as classes. Essa relaxação é apresentada no modelo $\mathcal{F}_q$.

No modelo $\mathcal{F}_q$, usamos uma cor referência denominada com c', cujo número de elementos cobertos é armazenado na variável de decisão $\beta \in \mathbb{Z}^+$, com a Restrição (2e). A partir do valor de β, os limites nos tamanhos das demais classes de cores são definidos nas Restrições (2f) e (2g).

$$\mathcal{F}_q: \quad \max \quad \sum_{j=1}^{m} w(u_j)x_j \tag{2a}$$

$$\text{s. a.} \qquad x_j \leq \sum_{u_j \in \mathcal{S}_\ell} y_\ell \qquad \forall\, u_j \in \mathcal{U} \tag{2b}$$

$$y_\ell \leq x_j \qquad \forall\, u_j \in \mathcal{S}_\ell, \forall\, \mathcal{S}_\ell \in \mathcal{S} \tag{2c}$$

$$\sum_{\ell=1}^{n} y_\ell = k \tag{2d}$$

$$\sum_{u_j \in C_{c'}} x_j = \beta \tag{2e}$$

$$\sum_{u_j \in C_d} x_j \leq \beta + \left\lfloor \frac{\tau}{2} \right\rfloor \qquad \forall\, d \in \mathcal{C},\ d \neq c' \tag{2f}$$

$$\sum_{u_j \in C_d} x_j \geq \beta - \left\lceil \frac{\tau}{2} \right\rceil \qquad \forall\, d \in \mathcal{C},\ d \neq c' \tag{2g}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in \{1,2,\ldots,m\} \tag{2h}$$

$$y_\ell \in \{0,1\} \qquad \forall\, \ell \in \{1,2,\ldots,n\} \tag{2i}$$

$$\beta \in \mathbb{Z}^+ \qquad \forall\, \ell \in \{1,2,\ldots,n\} \tag{2j}$$

$$\tag{2k}$$

Nos modelos $\mathcal{F}$ e $\mathcal{F}_q$ o conceito de justiça é tratado como um requisito de uma solução. Desse modo, toda solução deve atender a esse requisito, ou na igualdade ou com uma pequena folga (como é o caso da versão apresentada no modelo $\mathcal{F}_q$). Além disso, o objetivo é maximizar os ganhos com a soma dos pesos dos elementos cobertos.

Considerando ainda o caso quando o critério de justiça pode ser relaxado, um valor ideal τ pode ser difícil de definir. Desse modo, podemos relaxar o problema e, ainda assim, continuar com muitas instâncias inviáveis. Outra forma de relaxar essa restrição é considerar a justiça como parte do objetivo a ser alcançado, e não como um requisito de uma solução, de modo que adicionamos ao objetivo do problema obter o máximo de justiça possível. Isso faz com que o problema passe a ter dois objetivos a serem maximizados: a justiça e os ganhos.

Dado o conceito de justiça onde uma solução é justa se temos uma participação igual de todas as classes na cobertura, podemos definir a injustiça como a diferença entre os tamanhos das classes do cores cobertas. Desse modo, maximizar a justiça é equivalente a minimizar a injustiça, dentro do nosso modelo. A seguir, apresentamos o modelo $\mathcal{F}_d$ que representa uma nova versão do problema de justiça.

Adicionamos ao modelo $\mathcal{F}_d$ uma variável de decisão $z_{c,d} \in \mathbb{Z}$, que indica a diferença entre os tamanhos das classes c e d na cobertura. As Restrições (3b), (3c) e (3d) são as mesmas dos modelos anteriores e garantem que a solução é uma k-cobertura. Já a restrição descrita pelas Restrições (3e) são responsáveis por armazenar as diferenças entre os tamanhos das classes cobertas. As Restrições (3f), (3g) e (3h) definem os tipos das variáveis de decisão.

$$\mathcal{F}_d: \quad \max \quad \sum_{j=1}^{m} w(u_j)x_j - \sum_{c=1}^{\chi}\sum_{d=1}^{\chi} z_{c,d} \tag{3a}$$

$$\text{s. a.} \qquad x_j \leq \sum_{u_j \in \mathcal{S}_\ell} y_\ell \qquad \forall\, j \in \{1, 2, \ldots, m\} \tag{3b}$$

$$y_\ell \leq x_j \qquad \forall\, u_j \in \mathcal{S}_\ell, \forall\, \mathcal{S}_\ell \in \mathcal{S} \tag{3c}$$

$$\sum_{\ell=1}^{n} y_\ell = k \tag{3d}$$

$$\sum_{u_j \in C_c} x_j - \sum_{u_i \in C_d} x_i \leq z_{c,d} \qquad \forall\, c, d \in \mathcal{C} \tag{3e}$$

$$x_j \in \{0, 1\} \qquad \forall\, j \in \{1, 2, \ldots, m\} \tag{3f}$$

$$y_\ell \in \{0, 1\} \qquad \forall\, \ell \in \{1, 2, \ldots, n\} \tag{3g}$$

$$z_{c,d} \geq 0 \qquad \forall\, c, d \in \mathcal{C} \tag{3h}$$

Outro caminho possível para aumentarmos o espaço de soluções acontece quando a aplicação permite que o orçamento seja flexível, mas ainda precisamos atender ao critério de justiça à risca. Nesse caso, podemos remover a restrição que define o tamanho da cobertura e penalizar a discrepância entre o gasto real e o orçamento estipulado pela constante k. Isso leva a uma outra formulação para o problema de cobertura justa, descrito no modelo $\mathcal{F}_b$. Nesse modelo uma solução é uma cobertura justa, mas que pode ter um tamanho diferente daquele estipulado pelo orçamento k.

Com base no modelo $\mathcal{F}$, que é referente ao problema original, adicionamos uma variável de decisão $D \in \mathbb{Z}^+$ que armazenará a diferença entre o tamanho da cobertura e o orçamento da instância. Esse valor D é, então, penalizado na função objetivo com um multiplicador $\alpha \in \mathbb{R}^+$. A nova função objetivo é definida pela Restrição (4a).

O modelo $\mathcal{F}_b$ recebe dois novos conjuntos de restrições que são usados para definir os valores de D com base na cobertura. As Restrições (4d) e (4e) funcionam em conjunto para definir o valor de D usando uma diferença absoluta. Note que D é positivo e é adicionado na função objetivo como um fator negativo.

$$\mathcal{F}_b: \quad \max \quad \sum_{j=1}^{m} w(u_j)x_j - \alpha D \tag{4a}$$

$$\text{s. a.} \qquad x_j \leq \sum_{u_j \in \mathcal{S}_\ell} y_\ell \quad \forall\, u_j \in \mathcal{U} \tag{4b}$$

$$y_\ell \leq x_j \qquad \forall\, u_j \in \mathcal{S}_\ell, \forall\, \mathcal{S}_\ell \in \mathcal{S} \tag{4c}$$

$$\sum_{\ell=1}^{n} y_\ell - k \leq D \tag{4d}$$

$$k - \sum_{\ell=1}^{n} y_\ell \leq D \tag{4e}$$

$$\sum_{u_j \in C_c} x_j = \sum_{u_i \in C_d} x_i \quad \forall\, c, d \in \mathcal{C},\ c < d \tag{4f}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in \{1,2,\ldots,m\} \tag{4g}$$

$$y_\ell \in \{0,1\} \qquad \forall\, \ell \in \{1,2,\ldots,n\} \tag{4h}$$

$$D \in \mathbb{Z}^+ \qquad \forall\, \ell \in \{1,2,\ldots,n\} \tag{4i}$$

7.2 Justiça como Proporcionalidade

Os problemas da seção anterior consideram como justiça a igualdade no número de elementos cobertos de cada classe. Contudo, esse conceito de justiça pode não ser adequado para

todas as situações. Suponha que temos uma instância do problema com apenas duas classes. Na primeira classe se encontram 90% dos elementos e na segunda os 10% restantes. Considerando a justiça como igualdade, impomos um limite no tamanho total da cobertura com base na menor classe, de modo que poderíamos cobrir no máximo 20% do total de elementos do conjunto universo. Além disso, permitimos uma discrepância no percentual de cobertura de cada classe, pois todos os elementos da classe menor serão potencialmente cobertos enquanto da classe maior, no máximo 11,1% dos elementos poderiam ser cobertos.

Considerando esse caso de classes desbalanceadas, Asudeh *et al.* [3] definiu um novo conceito de justiça que especifica as proporções de cada classe que deve fazer parte da cobertura, isto é, junto com a entrada especificamos qual a distribuição de classes dos elementos da cobertura. Chamamos essa definição de justiça como justiça proporcional. Podemos usar uma distribuição proporcional à da instância, de modo que uma solução tenha a mesma distribuição de classes que o conjunto universo. No exemplo supracitado, para que uma solução seja justa, 90% deve ser composta de elementos da classe maior e 10% da classe menor.

Esse conceito é mais flexível, uma vez que podemos usar critérios diferentes para criar as distribuições. Isto é, seguir a proporção da instância é apenas uma das maneira de implementar o conceito de justiça proporcional. Suponha que temos um conjunto universo onde é sabido que uma classe está sub-representada na instância. Podemos favorecê-la com uma proporção maior do que a da entrada para tentar mitigar possíveis problemas de viés.

Assim como na seção anterior, podemos definir modelos de Programação Linear Inteira para essa interpretação do problema. Para isso, é preciso definir apenas um conjunto de constantes q_c que passam a ser parte da entrada representando a proporção desejada de cada classe de cor. O modelo $\mathcal{F}^P$ difere do $\mathcal{F}$ apenas nas Restrições (5e) que tratam do conceito de justiça, onde temos que o número de elementos cobertos com uma cor c deve ser igual à porcentagem q_c do total de elementos cobertos.

$$\mathcal{F}^P: \quad \max \quad \sum_{j=1}^{m} w(u_j)x_j \tag{5a}$$

$$\text{s. a.} \qquad x_j \leq \sum_{u_j \in \mathcal{S}_\ell} y_\ell \quad \forall\, j \in \{1,2,\ldots,m\} \tag{5b}$$

$$y_\ell \leq x_j \qquad \forall\, \ell \in \{1,2,\ldots,n\}, \forall\, j \in \{1,2,\ldots,m \mid u_j \in \mathcal{S}_\ell\} \tag{5c}$$

$$\sum_{\ell=1}^{n} y_\ell = k \tag{5d}$$

$$\sum_{u_j \in C_c} x_j = q_c \sum_{j=1}^{m} x_j \quad \forall\, c \in \mathcal{C} \tag{5e}$$

$$x_j \in \{0,1\} \qquad \forall\, j \in \{1,2,\ldots,m\} \tag{5f}$$

$$y_\ell \in \{0,1\} \qquad \forall\, \ell \in \{1,2,\ldots,n\} \tag{5g}$$

Usando o conceito de justiça como proporcionalidade, ainda podemos relaxar os conceitos de justiça e de orçamento da cobertura de forma análoga ao que é feito com os modelos $\mathcal{F}_q$ e $\mathcal{F}_b$ apenas substituindo as restrições que dizem respeito aos critérios de justiça. Ademais podemos incorporar esse conceito de justiça como um objetivo, análogo ao que é feito no modelo $\mathcal{F}_d$. Nesse caso, parte do objetivo não seria minimizar a diferença entre os tamanhos das classes e sim a diferença entre o tamanho de uma classe e seu valor estipulado na proporção dada na entrada.

8 Resultados Computacionais

Nesta seção detalhamos experimentos preliminares realizados. O objetivo desses experimentos é analisar os limites dos modelos apresentados na Seção 7 e, a partir desses limites, trabalhar em melhorias para que os modelos possam ser usados com instâncias reais.

Os experimentos apresentados a seguir foram executados em máquinas com processador Intel® Core™ i7-3770 com 8 núcleos de 3.4GHz, 32GB de memória RAM e sistema operacional Ubuntu, versão 18.04.2 LTS. Como *software* resolvedor do modelo, usamos o IBM CPLEX Studio, versão 12.8, com a função de paralelismo desativada. Implementamos os modelos mencionados na Seção 7 utilizando a linguagem de programação `C++` e compilados usando `g++` (versão 7.5), com as *flags* `C++19` e `-O3`.

Na seção a seguir detalhamos a criação dos diferentes tipos de instâncias artificiais criadas para os experimentos.

8.1 Criação de Instâncias Artificiais

Para gerarmos uma instância para o FMC precisamos definir os elementos do conjunto universo, a distribuição dos elementos em subconjuntos, os pesos de cada elemento, a quantidade de classes, a distribuição de elementos em cada classe e o valor do orçamento, isto é, o valor k que limita o tamanho da cobertura. Nesta seção, buscamos criar conjuntos de instâncias nos quais esses parâmetros possam ser controlados.

As instâncias geradas são baseadas em grafos aleatórios Erdös-Rényi [10]. Decidimos usar grafos como base das instâncias pois trazem um controle maior sobre a frequência dos elementos

em subconjuntos e número de elementos por subconjuntos.

Geramos 5 grafos aleatórios para cada cada par (n, p), com $n \in \{10, 20, \ldots, 100\}$ sendo o número de vértices e $p \in \{0.1, 0.2, \ldots, 0.5\}$ sendo a probabilidade de uma aresta ser adicionada, totalizando 250 grafos. Nesta etapa do projeto, fixamos o peso de uma aresta em 1. Decidimos usar apenas duas cores nas instâncias, isto é $\chi = 2$, que serão referenciadas como cor 1 e cor 2 no decorrer desta seção. Em seguida criamos 5 colorações aleatórias para as arestas de cada grafo, totalizando 1250 instâncias base. Nessas colorações, a primeira classe de cor tem $\lceil \frac{m}{2} \rceil$ arestas e a segunda classe tem $\lfloor \frac{m}{2} \rfloor$.

Para completar as instâncias precisamos definir um valor para k, isto é, definir o tamanho da subfamília selecionada. Para isso, executamos uma versão do modelo $\mathcal{F}$, definido pelas Restrições (1a) – (1g). Nesse caso, removemos a restrição do orçamento, Restrição (1d), que define que uma cobertura deve conter exatamente k subconjuntos (ou vértices) e maximizamos o número de subconjuntos selecionados mudando a função objetivo para a Inequação (6).

$$\text{maximize} \quad \sum_{\ell=1}^{n} y_\ell \tag{6}$$

Ao resolvermos o modelo definido pelas Restrições (6), (1b), (1c), (1e)–(1g) obtemos um valor para k. Caso o valor da solução desse modelo seja maior que zero, obtemos um valor válido para o orçamento k e geramos uma instância que possui pelo menos uma solução viável. Nos referimos a esse modelo como FMC-K.

Da forma como definimos os pesos das arestas e a distribuição de arestas nas duas classes de cor, uma cobertura justa máxima tende a conter todos os vértices do grafo, exceto quando o número de arestas é ímpar. Assim, usamos essas primeiras instâncias para verificação dos modelos. Primeiramente, executamos o modelo FMC-K com limite de tempo de execução de 15 minutos. Entretanto, esse limite não foi atingido por nenhuma das instâncias. Como era esperado, os valores de k encontrados nesse experimento foram próximos do número de vértices, logo, ao executarmos o modelo $\mathcal{F}$, todas as instâncias foram resolvidas rapidamente.

Com estes resultados, decidimos criar colorações um pouco mais criteriosas. Mas antes disso, definimos diferentes proporcionalidades de cada classe de cor: $(55, 45), (57.5, 42.5), (60, 40), (62.5, 37.5)$. Em cada lista de proporcionalidades temos dois números, o primeiro representa a porcentagem de elementos cobertos com a cor 1 e o segundo representa a porcentagem dos elementos com a cor 2. Para denotar cada par de proporcionalidade usamos $\mathcal{D}_x$, onde x é a diferença entre as duas porcentagens, assim o par $(55, 45)$ é representado por $\mathcal{D}_{10}$, por exemplo. Note que o conjunto inicialmente descrito possui proporcionalidade

$(50, 50)$ e é representado por $\mathcal{D}_0$. Note que o valor da proporcionalidade D_x também pode ser considerado como um índice de desbalanceamento da instância. A seguir descrevemos três modos de distribuição das cores para as arestas.

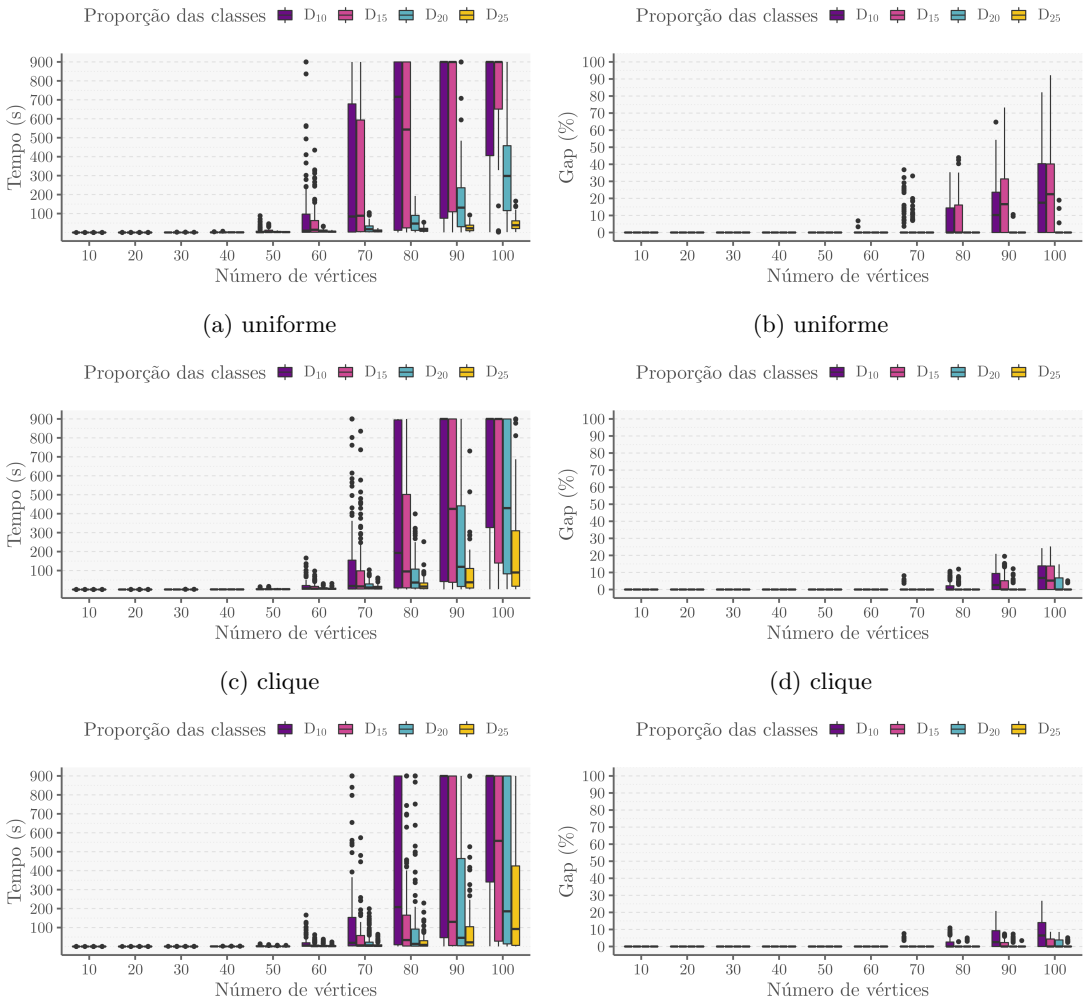
Nas instâncias do Tipo Uniforme, definimos a cor de uma aresta aleatoriamente mantendo a proporcionalidade. Nessas instâncias, cada vértice pode receber qualquer número de arestas com as cores 1 ou 2. Nas instâncias do Tipo Clique, identificamos, a partir de um vértice aleatório, cliques maximais nos grafos a serem coloridas com a cor 1, até que o número de elementos com a cor 1 seja igual ao estipulado pela proporção das classes. Nos casos em que colorir todas as arestas ultrapasse o limite no tamanho da classe de cor, apenas parte da clique receberá a cor 1. As arestas restantes, são coloridas com a cor 2.

A ideia por trás desses conjuntos é criar diferentes níveis de segregação dentro de um vértice. Espera-se que o número de arestas seja igual ou muito próximo nas instâncias uniformes. Já nas instâncias do tipo clique, haverá um desequilíbrio entre as classes dada a preferência pela formação de agrupamentos de aresta da cor 1. Por último, nas instâncias do tipo BFS, teremos como regra que parte dos vértices terão apenas arestas com a cor 1. Este é o caso com maior nível de segregação entre as classes.

Por último, criamos as instâncias do Tipo BFS (tipo 3). Nessas instâncias, usamos uma busca em largura (ou BFS do inglês *Breadth-First Search*) partindo de um vértice aleatório e colorindo todas as arestas encontradas com a cor 1 até que o número de arestas com a cor 1 seja igual ao estipulado pela proporção das classes. As demais arestas são coloridas com a cor 2. A ideia por trás desse conjunto de instâncias é obter um nível maior de segregação entre a representação das classes de cores de um subconjunto, de modo que, obrigatoriamente, teremos vértices incidentes a arestas de apenas uma das classes de cores.

A seguir apresentamos uma análise sobre a execução do modelo que determina um valor para o orçamento k nas instâncias artificiais. Nesses experimentos, consideramos o conceito de justiça como igualdade entre as classes de cores. Utilizamos também um tempo limite de 15 minutos para a execução de cada instância. Para cada um dos tipos de colorações citados acima e para cada D_x com $x \in \{10, 15, 20, 25\}$ criamos cinco colorações para cada um dos 250 grafos aleatórios gerados anteriormente. Assim, temos um total de 5000 instâncias de cada tipo de coloração.

Na Figura 5 apresentamos à esquerda a distribuição do tempo usado na execução do modelo FMC-K com limite de 900 segundos, e à direita os *gaps* indicando a diferença entre a melhor solução inteira encontrada e a limitante no momento em que a execução foi interrompida. Apresentamos um gráfico de tempo e um gráfico dos *gaps* para cada tipo de coloração. Em cada



(a) uniforme (b) uniforme

(c) clique (d) clique

(e) BFS (f) BFS

Figura 5: Distribuição dos tempos de execução e respectivos *gaps* do modelo FMC-K com os diferentes tipos de coloração.

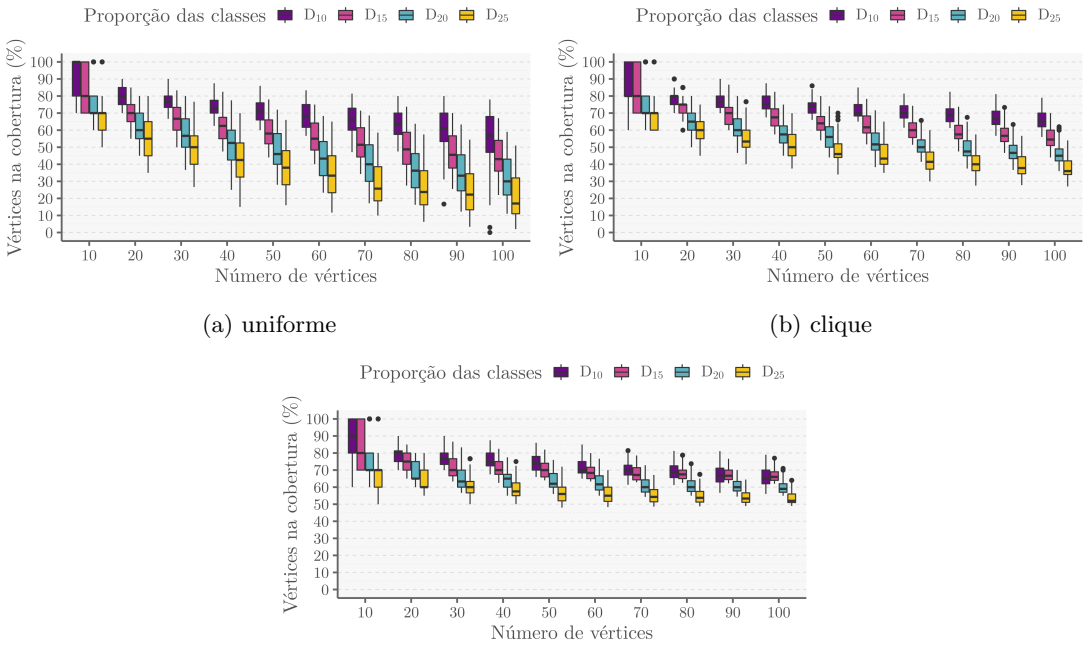
gráfico, separamos as instâncias por proporção de classe de cor, e representamos no eixo x o número de vértices da instância, ou seja, o número de subconjuntos da família $\mathcal{S}$.

As Figuras 5(a) e 5(b) referem-se às instâncias com coloração do tipo uniforme. No gráfico da Figura 5(a), vemos que para as instâncias com poucos vértices ($n \leq 60$) foi encontrado um valor ótimo muito rápido, o que também é refletido no gráfico da Figura 5(b), onde o *gap* de quase todas essas instâncias é zero.

Com esses gráficos, também podemos notar o impacto que o desbalanceamento tem no tempo de execução. Instâncias do tipo D_{20} e D_{25} possuem um alto grau de desbalanceamento e isso parece ter limitado espaço de busca, fazendo com que a execução do modelo seja mais rápida. Esta é uma observação interessante, pois temos de experimentos anteriores que as instâncias com grau zero de desbalanceamento também foram resolvidas dentro do tempo limite. Isso pode indicar que graus de desbalanceamento na coloração que são mais sutis podem resultar em instâncias mais difíceis de serem resolvidas pelo modelo de programação linear inicialmente proposto por Asudeh *et al.* [3].

Nas Figuras 5(c) e 5(d) e Figuras 5(e) e 5(f) temos os resultados de tempo de execução e *gap* para as instâncias com colorações dos tipos clique e BFS, respectivamente. Esses gráficos apresentam resultados similares aos resultados das instâncias com a coloração do tipo uniforme. Contudo, essas instâncias obtiveram valores de *gap* mais baixos, tendo um número menor de instâncias cuja execução foram encerradas por limite de tempo. Contudo, esses dois conjuntos de instâncias tiveram um maior número de instâncias com grau de desbalanceamento D_{20} que precisaram de mais tempo para completar a execução. Isso sugere que esse nível mais alto de desbalanceamento combinado com a maior separação das classes também pode gerar instâncias mais difíceis de serem resolvidas.

Após a análise inicial dos resultados da execução do modelo FMC-K, compilamos na Figura 6 a porcentagem dos vértices pertencentes às coberturas nas soluções encontradas, isto é, o tamanho do orçamento k em relação ao número de subconjuntos da família $\mathcal{S}$. Para todos tipos de coloração, temos que quanto maior o nível de desbalanceamento menor o valor do orçamento. Nas instâncias com coloração uniforme, percebemos uma maior variação nos valores de orçamento, isso pode ser causado pelo fato de as soluções não serem provadas ótimas, sugerindo a existência de outras soluções com um orçamento maior.



(a) uniforme

(b) clique

(c) BFS

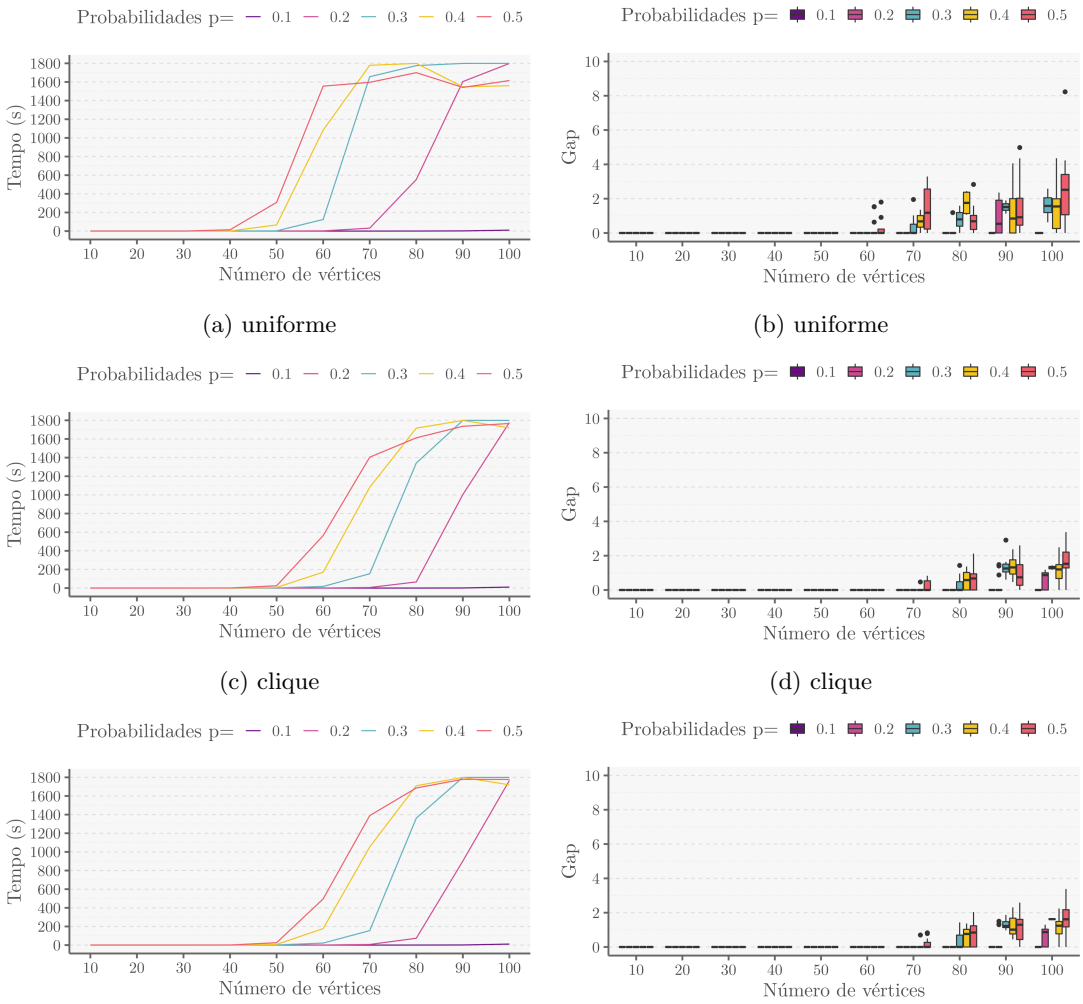
Figura 6: Porcentagem de vértices na cobertura após a execução do modelo FMC-K.

8.2 Justiça como Igualdade

Nos experimentos anteriores, realizamos o último passo para a criação de instâncias para o problema de cobertura justa que considera justiça como igualdade. A seguir, apresentamos os experimentos executados com tais instâncias e o modelo $\mathcal{F}$ proposto por Asudeh *et al.* [3]. Para as execuções a seguir consideramos apenas as instâncias com o menor nível de desbalanceamento (D_{10}) e aumentamos o limite de tempo de execução para 30 minutos, as demais configurações são as mesmas do experimento anterior.

Na Figura 7 apresentamos os tempos médios de execução e a distribuição dos valores de *gap*. Como nesse experimento usamos apenas um nível de desbalanceamento nas colorações, dividimos os valores nos gráficos com base apenas nos tamanhos das instâncias. No eixo x temos o número de vértices e as linhas nos gráficos de tempo representam os conjuntos de instâncias que foram gerados com o parâmetro p do grafo aleatório $G_{n,p}$. Assim, temos que quanto maior o valor de p, maior o número de arestas do grafo.

Houve um total de 491 instâncias (aproximadamente 13%, com números de vértices de entre 60 e 100) para as quais não foram encontradas uma solução com o modelo $\mathcal{F}$, o valor de orçamento definido pelos experimentos da seção anterior e o tempo limite de 30 minutos. Sendo 185 instâncias do conjunto uniforme, 152 do conjunto cliques e 154 do conjunto BFS. Os valores de *gaps* representados nas Figuras 7(b), 7(d) e 7(f) excluem as 491 instâncias citadas acima. Apesar



(a) uniforme (b) uniforme

(c) clique (d) clique

(e) BFS (f) BFS

Figura 7: Média dos tempos de execução e distribuição dos *gaps* do experimento com o modelo $\mathcal{F}$ e os diferentes tipos de coloração.

disso, temos valores baixos de *gap* para a maioria das instâncias.

As médias de tempo de execução nas Figuras 7(a), 7(c) e 7(e) apresentam pouca variação entre os tipos de coloração. Esses valores incluem as 491 instâncias citadas acima, e que influenciaram nos valores altos de tempo de execução. Considerando o número de instâncias para as quais não obtivemos solução nessa versão do problema, estudaremos em seguida como melhorar a modelagem, a fim de tornar o modelo viável para uso com instâncias reais. Além disso, esses resultados serão usados como *benchmark* para o desenvolvimento das demais abordagens propostas neste projeto.

Referências

[1] C. Aggarwal, ***Data Classification: Algorithms and Applications***, sér. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis, 2014, ISBN: 9781466586741.

[2] J. Angwin, J. Larson, S. Mattu e L. Kirchner, "Machine Bias", ***ProPublica***, 2016. endereço: `http://bit.ly/37t0w3C` (acesso em 25/01/2020).

[3] A. Asudeh, T. Berger-Wolf, B. DasGupta e A. Sidiropoulos, "Maximizing coverage while ensuring fairness: a tale of conflicting objective", pp. 1–38, 2020. arXiv preprint arXiv: `2007.08069`.

[4] T. Aykin, "Optimal shift scheduling with multiple break windows", ***Management Science***, v. 42, n. 4, pp. 591–602, 1996.

[5] A. Başar, B. Çatay e T. Ünlüyurt, "A taxonomy for emergency service station location problem", ***Optimization Letters***, v. 6, n. 6, pp. 1147–1160, 2012.

[6] F. Ben Rejab, K. Nouira e A. Trabelsi, "Health Monitoring Systems Using Machine Learning Techniques", em ***Intelligent Systems for Science and Information***. 2014, pp. 423–440.

[7] N. Bilal, P. Galinier e F. Guibault, "A new formulation of the set covering problem for metaheuristic approaches", ***International Scholarly Research Notices***, v. 2013, 10 pages, Article ID 203032, 2013.

[8] A. Christin, A. Rosenblat e D. Boyd, "Courts and predictive algorithms", em ***Data & CivilRight***, Washington, DC, out. de 2015. endereço: `https://bit.ly/3b6GTQ1`.

[9] V. Chvatal, "A greedy heuristic for the set-covering problem", ***Mathematics of Operations Research***, v. 4, n. 3, pp. 233–235, 1979.

[10] P. Erdős e A. Rényi, "On the evolution of random graphs", ***Publications of the Mathematical Institute of the Hungarian Academy of Sciences***, v. 5, n. 1, pp. 17–60, 1960.

[11] T. A. Feo e M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures", ***Journal of Global Optimization***, v. 6, n. 2, pp. 109–133, 1995.

[12] C. Guse, "Citi Bike neglects poor NYC neighborhoods and communities of color: report", ***Daily News***, 2019. endereço: `http://bit.ly/3aoNK7W` (acesso em 25/11/2020).

[13] R. Hassin e A. Levin, "A better-than-greedy approximation algorithm for the minimum set cover problem", ***SIAM Journal on Computing***, v. 35, n. 1, pp. 189–200, 2005.

[14] D. Ingold e S. Soper, "Amazon Doesn't Consider the Race of Its Customers. Should It?", ***Bloomberg***, 2016. endereço: `http://bloom.bg/3p0DHKz` (acesso em 25/01/2020).

[15] R. M. Karp, "Reducibility among Combinatorial Problems", em ***Complexity of Computer Computations***. Springer US, 1972, pp. 85–103, ISBN: 978-1-4684-2001-2.

[16] J. Kleinberg, J. Ludwig, S. Mullainathan e A. Rambachan, "Algorithmic Fairness", ***AEA Papers and Proceedings***, v. 108, pp. 22–27, 2018.

[17] J. Kritter, M. Brévilliers, J. Lepagnot e L. Idoumghar, "On the optimal placement of cameras for surveillance and the underlying set cover problem", ***Applied Soft Computing***, v. 74, pp. 133–153, 2019.

[18] J. M. Lanza-Gutierrez, N. C. Caballe, B. Crawford, R. Soto, J. A. Gomez-Pulido e F. Paredes, "Exploring Further Advantages in an Alternative Formulation for the Set Covering Problem", ***Mathematical Problems in Engineering***, v. 2020, Article ID 5473501, 2020.

[19] A. Levin, "Approximating the unweighted k-set cover problem: greedy meets local search", ***SIAM Journal on Discrete Mathematics***, v. 23, n. 1, pp. 251–264, 2009.

[20] Y. Lin, Y. Guan, A. Asudeh e H. V. J. Jagadish, "Identifying Insufficient Data Coverage in Databases with Multiple Relations", ***Proceedings of the VLDB Endowment***, v. 13, n. 12, pp. 2229–2242, 2020.

[21] R. Mac, "Facebook Apologizes After A.I. Puts 'Primates' Label on Video of Black Men", ***New York Times***, 2021. endereço: `https://nyti.ms/3FuVRfi` (acesso em 09/01/2022).

[22] N. Mladenović e P. Hansen, "Variable neighborhood search", ***Computers & Operations Research***, v. 24, n. 11, pp. 1097–1100, 1997.

[23] C. O'Neil, ***Algoritmos de Destruição em Massa***. Editora Rua do Sabão, 2021.

[24] R. Raz e S. Safra, "A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP", em ***Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'1997)***, ACM, 1997, pp. 475–484.

[25] Z.-G. Ren, Z.-R. Feng, L.-J. Ke e Z.-J. Zhang, "New ideas for applying ant colony optimization to the set covering problem", ***Computers & Industrial Engineering***, v. 58, n. 4, pp. 774–784, 2010.

[26] A. Rose, "Are Face-Detection Cameras Racist?", ***Time***, 2010. endereço: `https://bit.ly/3A7lIsC` (acesso em 14/01/2022).

[27] T. Silva, "Linha do Tempo do Racismo Algorítmico", ***Blog do Tarcízio Silva***, 2020. endereço: `http://bit.ly/2J1Zv9l` (acesso em 02/12/2020).

[28] C. M. Suelen Laurindo Ricardo Moraes e F. Vasques, "Assessment of Different Algorithms to Solve the Set-Covering Problem in a Relay Selection Technique", em ***Proceedings of the 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'2020)***, 2020, pp. 206–213.

[29] V. V. Vazirani, ***Approximation Algorithms***. Berlin, Germany: Springer-Verlag, 2001.

[30] G. Villavicencio, M. Valenzuela, F. Altimiras, P. Moraga e H. Pinto, "A K-Means Grasshopper Optimisation Algorithm Applied to the Set Covering Problem", em ***Proceedings of the 10th CSOC: Artificial Intelligence and Bioinspired Computational Methods (CSOC'2020)***, Springer International Publishing, 2020, pp. 312–323.

[31] A. Waters e R. Miikkulainen, "GRADE: Machine learning support for graduate admissions", ***AI Magazine***, v. 35, n. 1, pp. 64–64, 2014.