UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO MATEUS DOS ANJOS SILVA

UMA HEURÍSTICA PROBABILÍSTICA PARA O PROBLEMA DE CLASSIFICAÇÃO GEODÉSICA

QUIXADÁ

2021

FRANCISCO MATEUS DOS ANJOS SILVA

UMA HEURÍSTICA PROBABILÍSTICA PARA O PROBLEMA DE CLASSIFICAÇÃO GEODÉSICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Paulo Henrique Macêdo de Araújo

QUIXADÁ

2021

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

S58h Silva, Francisco Mateus dos Anjos.
Uma heurística probabilística para o problema de Classificação Geodésica / Francisco Mateus dos Anjos Silva. – 2021.
43 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2021.
Orientação: Prof. Dr. Paulo Henrique Macêdo de Araújo.

1. Geodésica-Classificação. 2. Algoritmos heurísticos. 3. Algoritmo probabilístico. I. Título.
CDD 004

FRANCISCO MATEUS DOS ANJOS SILVA

UMA HEURÍSTICA PROBABILÍSTICA PARA O PROBLEMA DE CLASSIFICAÇÃO GEODÉSICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: ___/___/____

BANCA EXAMINADORA

Prof. Dr. Paulo Henrique Macêdo de Araújo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fábio Carlos Sousa Dias
Universidade Federal do Ceará (UFC)

Prof. Dr. Wladimir Araújo Tavares
Universidade Federal do Ceará (UFC)

Dedico este trabalho primeiramente a Deus, por ter me abençoado para conseguir esta conquista e aos meus pais, Franciné e Erineide, por todos os incentivos. Esta vitória também é de vocês.

AGRADECIMENTOS

Agradeço a Deus por todas as bênçãos recebidas durante a graduação.

Aos meus pais Franciné e Erineide, por todo o esforço, sacrifício e apoio em todos esses anos que me permitiram chegar a este momento.

Ao meu irmão Franciné Júnior por desde sempre compartilhar os momentos de alegrias e dificuldades.

Ao meu Prof. Orientador Dr. Paulo Henrique Macêdo de Araújo pela ótima orientação e suporte no desenvolvimento do trabalho.

Aos meus professores da banca examinadora Wladimir Araújo e Fábio Dias pela contribuição com sugestões de melhorias para o trabalho.

A todos os professores da Universidade Federal do Ceará - Campus Quixadá por me proporcionar o conhecimento, pelo tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

A todos os meus amigos que contribuíram de alguma forma para eu concluir essa jornada, são eles: Carlos, Maurício, Marcelo, Robson, Laura, Sara e todos os outros não citados.

“O sucesso é a soma de pequenos esforços repetidos dia após dia.”

(Robert Collier)

RESUMO

Este trabalho propõe uma heurística probabilística para o problema de Classificação Geodésica de Grupo Único com 2 Classes (CGGU-2), que foi introduzido em Araújo *et al.* (2019a), onde os autores apresentam um modelo de programação linear inteira para resolvê-lo de forma exata. O problema consiste em classificar dados ao reconhecer padrões que foram definidos por convexidade geodésica a partir de informações pré-estabelecidas e representadas por uma relação binária através de um grafo. Realizamos experimentos computacionais para avaliar a eficiência computacional e a precisão de classificação da abordagem proposta, comparando-a com alguns métodos clássicos de solução para o problema de classificação. Vale salientar que nossa heurística é um aprimoramento da heurística desenvolvida para o problema de Classificação Geodésica de Grupos Múltiplos com 2 Classes em Vieira (2019).

Palavras-chave: Geodésica-Classificação. Algoritmos heurísticos. Algoritmo probabilístico.

ABSTRACT

This work proposes a probabilistic heuristic for the 2-Class Single Group Geodesic Classification Problem (2-SGGC), which was introduced in Araújo *et al.* (2019a), where the authors present an integer linear programming model to solve it exactly. The problem aims to classify data by recognizing patterns that were defined by geodesic convexity from pre-established information and represented by a binary relationship using a graph. We performed computational experiments to evaluate the computational efficiency and classification accuracy of the proposed approach, comparing it with some classic methods of solution for the classification problem. It is worth mentioning that our heuristic is an improvement of the heuristic developed for the 2-Class Multi Group Geodesic Classification problem in Vieira (2019).

Keywords: Geodesic-Classification. Heuristic algorithms. Probabilistic algorithm.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CGGU-2	Classificação Geodésica de Grupo Único com 2 Classes
CG	Classificação Geodésica
SVM	*Support Vector Machine*
MLP	*Multi Layer Perceptron*
CRIO	*Classification and Regression via Integer Optimization*
BFS	*Breadth First Search*
SPECT	*Single Proton Emission Computed Tomography*

SUMÁRIO

1 INTRODUÇÃO

Grandes quantidades de dados são gerados a cada instante no mundo inteiro por diversos tipos de pessoas, desde simples usuários de *internet* que geram um grande volume de informações sobre suas preferências e rejeições, a pesquisadores atuando em diferentes áreas de conhecimento. Sabendo disso, pode-se extrair diversas informações úteis desses dados gerados. Para isso, os dados são catalogados e classificados em grupos considerando suas similaridades em algum contexto para que seja possível prever similaridades entre amostras de informações desconhecidas. Esse procedimento constitui a chamada aprendizagem supervisionada ou *supervised learning*, que é uma ferramenta amplamente usada em muitas situações cotidianas para solucionar alguns dos grandes problemas da área de tecnologia da informação em *big data* e *data science* (ARAÚJO *et al.*, 2019a).

O procedimento de aprendizagem supervisionada é feito quando a partir de um conjunto de dados rotulados previamente definido deseja-se encontrar uma função que seja capaz de predizer os rótulos de conjuntos de dados ainda não rotulados. Tal procedimento possui duas fases: a fase de treinamento e a fase de predição. Na primeira fase, fase de treinamento, o conjunto de amostras é analisado e classificado de tal modo a agrupar amostras consideradas semelhantes em uma dada perspectiva considerada em uma mesma classe. Já na segunda fase, fase de predição, é determinada em qual classe uma amostra fora do conjunto inicialmente considerado se encaixaria melhor. Na fase de treinamento, surge um problema que chamamos de problema de classificação. Para tal problema, consideramos apenas duas classes neste trabalho.

Em Cortes e Vapnik (1995), os autores propuseram um método clássico para separar conjuntos de amostras que foram definidas como pontos em um espaço multidimensional. Este método é chamado de *Support Vector Machine* (SVM). O método consiste em separar o espaço de amostras com um hiperplano em uma dimensão maior, dividindo as duas classes em dois subespaços, onde cada um corresponde a uma classe. Portanto, quando um ponto se encontra fora do conjunto de amostras inicial terá sua classe prevista de acordo com o subespaço que ele pertence.

Fundamentado no método SVM e suas adaptações em Bertsimas e Shioda (2007) e Corrêa *et al.* (2019b), foi definido um problema de classificação em grafos análogo à versão Euclidiana do problema de classificação no espaço multidimensional. Esse novo problema definido em Araújo *et al.* (2019a) é chamado de problema de Classificação Geodésica (CG). Ele é definido fazendo uso da convexidade geodésica em grafos para agrupar amostras similares,

onde cada amostra é um vértice do grafo. Dessa forma, um par de vértices que possuem aresta entre si são considerados similares (no contexto da aplicação), e portanto podem representar uma relação binária entre as amostras.

Nesse contexto, o presente trabalho tem como objetivo desenvolver uma heurística probabilística para a obtenção de uma boa solução viável para o problema CG, cujas aplicações atendam às seguintes hipóteses (ARAÚJO *et al.*, 2019a):

- No conjunto de amostras existe um padrão que não se manifesta claramente e que pode ser representado através de uma noção de convexidade. Trata-se de um tema relativamente recente de investigação na Teoria dos Grafos que surgiu como uma analogia ao conceito de convexidade Euclidiana (analogia essa referente à distância entre pontos). Apesar de existir várias noções de convexidade em grafos, este trabalho está focado no conceito de convexidade geodésica, definido pelo menor caminho no grafo (PELAYO, 2013). Nesse caso, as amostras não precisam ser caracterizadas por valores numéricos. Com isso, o problema de classificação torna-se puramente combinatório. Entretanto, de acordo com os trabalhos encontrados na literatura, a inspiração para este trabalho são métodos de resolução para o caso da classificação através da convexidade Euclidiana. No caso Euclidiano, em um espaço multidimensional, as amostras são expressas por vetores numéricos, e o padrão está de acordo com a posição dessas amostras. O estudo do uso de convexidade geodésica em grafos tem duas motivações principais. A primeira motivação é de interesse teórico, no qual se refere à possibilidade de estabelecer novos problemas em grafos que possam trazer contribuição para métodos de resolução, inclusive àqueles baseados na convexidade Euclidiana. A segunda motivação tem um propósito mais prático que resume-se a permitir que o padrão seja expresso por alguma relação binária a partir de um conjunto de amostras.
- O conjunto de dados pode conter alguns pontos discrepantes, chamados de *outliers*, resultante de possíveis erros de amostragem ou devido a características do fenômeno que está sendo estudado. Do ponto de vista da modelagem matemática, os pontos discrepantes são aqueles que se desviam do padrão implícito do conjunto de amostras de sua classe. A possível ocorrência de discrepantes gera um desafio adicional, pois eles precisam ser identificados e desconsiderados do processo de determinação do padrão das amostras. É preciso ter bastante atenção com o processo de eliminação de pontos discrepantes pois pode ocorrer a eliminação de alguns pontos não discrepantes.

Considerando as hipóteses acima, o problema CG abordado neste trabalho se torna um problema de otimização matemática que consiste em separar o espaço de amostras em conjuntos convexos e associar cada conjunto à sua classe de similaridade correspondente. Tal associação caracteriza o padrão que é usado para predizer a classe de dados desconhecidos. Geralmente, o modo mais usado e mais fácil de classificar usando essa abordagem é a separação linear, usada, por exemplo, no método SVM. Sendo assim, o problema CG faz uma analogia desse método para o contexto de grafos. O mesmo é considerado um problema de otimização combinatória, onde queremos encontrar a melhor solução dentre várias possíveis, ou seja, que satisfazem as restrições do problema.

O problema de classificação tem aplicações em diversas áreas, como reconhecimento de imagem, estatísticas clássicas e mineração dados. Alguns exemplos são: conteúdo de recomendação do Netflix, classificação de texto do Twitter, filtragem de *spam* para *e-mails* e detecção de comunidades em redes sociais. No conteúdo de recomendação do Netflix, por exemplo, as informações são capturados em cada acesso ao serviço em questão e usadas para atualizar constantemente os algoritmos e aperfeiçoar a previsão feita sobre o que usuário provavelmente vai querer assistir. Os dados, algoritmos e sistemas alimentam uns aos outros para produzir novas recomendações.

O presente trabalho está organizado da seguinte forma. Na Seção 2 são discutidos os trabalhos relacionados ao trabalho proposto. A Seção 3 trata da fundamentação teórica, onde apresentamos notações e definimos o problema de CGGU-2. Posteriormente, a Seção 4 aborda os procedimentos metodológicos. Os experimentos computacionais e os resultados são mostrados na Seção 5. Por fim, na Seção 6 apresentamos as conclusões e os trabalhos futuros.

2 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados a este.

2.1 *Classification and regression via integer optimization*

Em Bertsimas e Shioda (2007), motivados pelos avanços significativos na otimização inteira na última década, os autores propõem métodos de otimização baseados em programação inteira mista para os problemas estatísticos clássicos de classificação e regressão. Os problemas de classificação são aqueles que procuram predizer a classe de um dado dentro do escopo limitado das possibilidades existentes. Por exemplo, esta classe pode ser se uma pessoa está gripada ou não, se a pessoa é do sexo masculino ou feminino, sendo que nestes casos ou a previsão será uma ou outra. E os problemas de regressão são aqueles em que queremos predizer um valor numérico específico. Tal valor pode ser o preço de um carro, idade de uma pessoa, ou algo assim.

Os autores apresentam o pacote de software chamado *Classification and Regression via Integer Optimization* (CRIO). Este pacote separa os pontos de dados em diferentes regiões poliédricas. Na classificação, cada região recebe uma classe, enquanto na regressão cada região possui seus próprios coeficientes de regressão distintos. Com isso, o CRIO demonstra um potencial de impacto significativo dos métodos de otimização em problemas estatísticos computacionais e mineração de dados.

2.2 *On the combinatorics of the 2-class classification problem*

No trabalho de Corrêa *et al.* (2019b), foram explorados os aspectos combinatórios do problema de classificação com convexidade Euclidiana por meio de formulações de programação inteira mista baseadas no pacote CRIO. De forma mais detalhada, foi feito um estudo poliédrico de um problema de classificação específico de duas classes, solicitando uma classificação de pontos em grupos linearmente separáveis, e fazendo com que os valores *outliers* não sejam classificados em nenhum grupo. Esses valores *outliers* são valores que não representam bem o padrão da classe que eles pertencem. Portanto, o principal interesse dos autores foi explorar a estrutura poliédrica para uma formulação de programação inteira para a versão Euclidiana do problema de Classificação.

2.3 *The geodesic classification problem on graphs*

Em Araújo *et al.* (2019a), os autores apresentaram um modelo de programação linear inteira para o problema CG de forma análoga à versão do problema de classificação baseado nos conceitos de convexidade Euclidiana discutidos em Corrêa *et al.* (2019a). Este modelo apresentado consiste em classificar dados ao reconhecer padrões que foram definidos por convexidade geodésica a partir de informações pré-estabelecidas e representadas por uma relação binária através de um grafo. Portanto, foram feitos experimentos computacionais, a fim de avaliar a eficiência da otimização combinatória e precisão de classificação da abordagem proposta. A versão do problema de Classificação apresentada no artigo de Araújo *et al.* (2019a) restringe o uso de apenas um grupo para cada uma das duas classes. Portanto, chamaremos tal problema de problema de Classificação Geodésica de Grupo Único com 2 Classes (CGGU-2).

No artigo Araújo *et al.* (2019a) foi formulado um modelo de programação inteira para esse problema de CG. Para descrever a formulação inteira, foram definidas algumas notações. Foi denotado por $T_K(v)$ o conjunto $\{S \subseteq V_K : v \in H[S] \setminus S\}$, para $K \in \{B,R\}$ e $|S| \geq 2$, onde $H[S]$ é o fecho convexo de S e B, R se referem às duas classes do problema. $T_K(v)$ significa a coleção de subconjuntos de vértices inicialmente classificados com a classe K tais que o fecho convexo alcança o vértice $v \in V$.

Dois tipos de variáveis binárias foram definidas. Para cada $v \in V_{BR}$, $a_v = 1$ significa que o vértice $v \in V_B$ (respectivamente $v \in V_R$) pertence ao grupo A_B (resp. A_R), e $a_v = 0$ se v não pertence ao grupo. Por outro lado, para cada $v \in V_N$, p_v indica a classe de v como segue: se $v \in H[A_B]$ (resp. $v \in H[A_R]$), então $p_v = 0$ (resp. $p_v = 1$); caso v não pertença ao fecho convexo de nenhum dos grupos, então p_v pode ser definido arbitrariamente. Portanto, a formulação tem como objetivo maximizar $\sum_{v \in V_{BR}} a_v$, o que equivale a minimizar a quantidade de *outliers*, sujeito às seguintes restrições:

$$a_v + \sum_{w \in S} a_w \leq |S|, \qquad (v \in V_B, S \in T_R(v)) \; ou \; (v \in V_R, S \in T_B(v)) \tag{2.1}$$

$$p_v + \sum_{w \in S} a_w \leq |S|, \qquad v \in V_N, S \in T_B(v) \tag{2.2}$$

$$-p_v + \sum_{w \in S} a_w \leq |S| - 1, \qquad v \in V_N, S \in T_R(v) \tag{2.3}$$

$$a_v, \; p_u \in \{0,1\}, \qquad v \in V_{BR}, u \in V_N \tag{2.4}$$

A restrição (2.1) é análoga à desigualdade de inclusão convexa definida em Corrêa *et al.* (2019b) no caso de grupo único. Ou seja, a restrição (2.1) diz que se $v \in V_{BR}$ pertence ao

fecho convexo de um subconjunto de vértices inicialmente classificados da classe oposta à de v, então pelo menos um desses vértices (incluindo v) deve ser *outlier*. Isto garante que $v \in V_{BR}$ se torne um *outlier* sempre que pertencer ao fecho convexo de um conjunto S de amostras da classe oposta. As restrições de cobertura são definidas por (2.2) e (2.3), onde implicam que se $v \in V_N \cap H[A_B]$ (resp. $v \in V_N \cap H[A_R]$), então p_v deve receber valor 0 (resp. 1). Portanto, como a variável $p_v \in \{0,1\}$, v não pode estar no fecho convexo de ambos os grupos de classes opostas simultaneamente (ARAÚJO *et al.*, 2019a).

2.4 Trabalho proposto

Neste trabalho, desenvolvemos uma heurística probabilística para o problema de CGGU-2. A heurística proposta aqui será baseado em ideias e conceitos apresentados nos trabalhos citados anteriormente, de onde será moldado para encontrar boas soluções para o problema de CGGU-2.

A Tabela 1 mostra as semelhanças e as diferenças dos trabalhos citados neste capítulo com o trabalho proposto.

Tabela 1 – Tabela demonstrando as semelhanças entre os trabalhos.

Semelhanças	Trabalhos			
	Bertsimas e Shioda (2007)	Corrêa *et al.* (2019b)	Araújo *et al.* (2019a)	Trabalho Proposto
Programação Inteira Mista	Sim	Sim	Sim	Não
Programação Quadrática	Sim	Não	Não	Não
Classificação Euclidiana	Sim	Sim	Não	Não
Classificação Geodésica	Não	Não	Sim	Sim
Heurística Probabilística	Não	Não	Não	Sim

Fonte: Elaborada pelo autor.

Analisando a Tabela 1, percebemos que os três trabalhos relacionados usam programação inteira mista, mas somente o trabalho de Bertsimas e Shioda (2007) usa também programação quadrática. Vemos que os trabalhos de Corrêa *et al.* (2019b) e de Bertsimas e Shioda (2007) tratam do problema de classificação Euclidiana, já os trabalhos de Araújo *et al.* (2019a) e o presente trabalho tratam do problema de classificação Geodésica. Por fim, podemos perceber que apenas nosso trabalho aborda o problema de classificação de maneira heurística e com o uso de um certo grau de aleatoriedade. Além disso, vale salientar que nossa heurística é um aprimoramento da heurística desenvolvida para o problema de Classificação Geodésica de Grupos Múltiplos com 2 Classes em Vieira (2019).

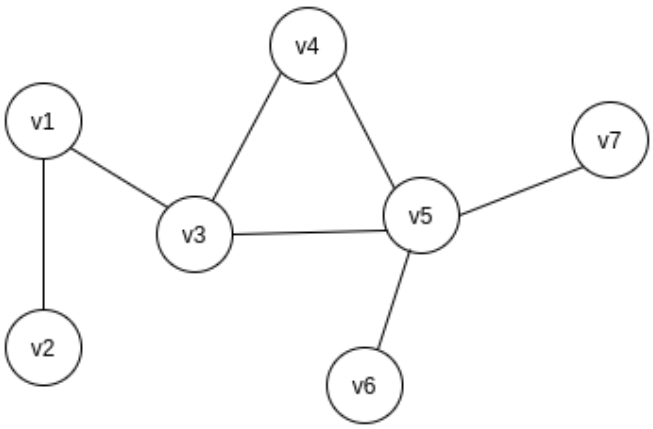
3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos de Grafos, o problema de Classificação Geodésica e os algoritmos probabilísticos, importantes para auxiliar no entendimento do trabalho.

3.1 Conceitos de Grafos

Pela notação em Pelayo (2013), um grafo não-direcionado G é um par ordenado $G = (V, E)$, composto por um conjunto finito $V = \{1, 2, ..., n\}$, cujos elementos são chamados de vértices, e por um conjunto de pares não ordenados $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$, cujos elementos são chamados de arestas. Denotamos por $n = \mid V \mid$ e $m = \mid E \mid$ a cardinalidade de cada um desses conjuntos. Dois vértices $u, v \in V$ são adjacentes (ou vizinhos) se $\{u, v\} \in E$, e não adjacentes caso contrário. Duas arestas $\{e, f\} \in E$ são adjacentes se $e \cap f \neq \varnothing$, e não adjacente caso contrário. O grau de um vértice $v \in V$, denotado por $deg(v)$, corresponde ao número de vizinhos de v em G, isto é $\mid \{u : \{v, u\} \in E\} \mid$. A densidade de G, denotada por $d(G)$, é dada por $d(G) = \frac{2m}{n(n-1)}$. $\delta(G)$ denota o grau do vértice de menor grau em G e $\Delta(G)$ o correspondente em grau máximo. A Figura 1 representa um grafo G não-direcionado com $\Delta(G) = 4$, pois v_5 é o vértice com maior grau, isto é, ele tem 4 vizinhos, e com $\delta(G) = 1$, pois v_2, v_6 e v_7 são os vértices com menor grau, cada um com 1 vizinho. Podemos observar também que o grafo tem $n = 7$ e $m = 7$, portanto tem $d(G) = \frac{1}{3}$.

Figura 1 – Exemplo de grau e de densidade de um grafo.



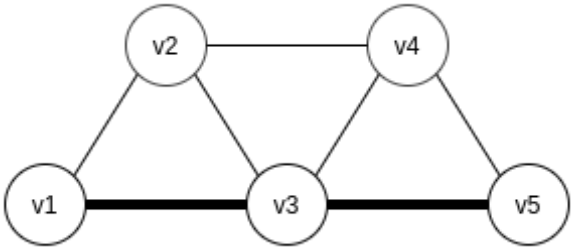
Fonte: Elaborada pelo autor.

Em um grafo G, um caminho é uma sequência de vértices distintos $P = < v_1, v_2, ..., v_l >$, tal que $\{v_i, v_{i+1}\} \in E(G)$ para $i = 1, ..., l-1$. Neste caso, denotamos $V(P) = \{v_1, ..., v_l\}$ e $E(P) = \{\{v_i, v_{i+1}\} : i = 1, ..., l-1\}$. Chamamos v_1 de início e v_l de final de P, e dizemos que P

é um caminho de v_1 a v_l. O tamanho de P é igual a $\mid E(P) \mid = \mid V(P) \mid -1$. Se o tamanho de P tiver o menor valor dentre todos os caminhos de v_1 até v_l, então P é um caminho mínimo.

A Figura 2 representa um grafo G não-direcionado com P sendo um caminho mínimo $P =< v_1, v_3, v_5 >$, pois P é o menor caminho dentre todos os outros caminhos de v_1 até v_5.

Figura 2 – Exemplo de um caminho mínimo $P =< v_1, v_3, v_5 >$.



Fonte: Elaborada pelo autor.

Um caminho mínimo entre v e w em G é também chamado de geodésico entre v e w no grafo. Um grafo G é conexo se existir um caminho entre qualquer par de vértices de G. A analogia entre o conceito de conjunto convexo em matemática contínua e discreta é feita considerando o conjunto de vértices de um grafo conexo e a distância entre dois vértices como um espaço métrico, onde esta distância é a quantidade de arestas em um caminho mínimo entre eles (ARTIGAS *et al.*, 2011). Dessa forma, um subconjunto de vértices S de $V(G)$ é convexo se contiver os vértices de todos os caminhos mínimos que conectam qualquer par de vértices em S.

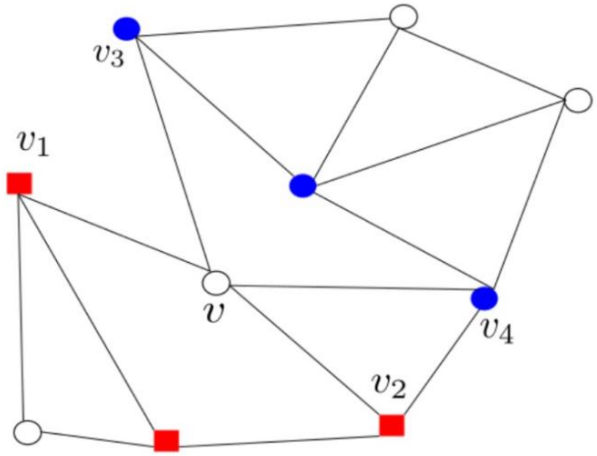
O intervalo fechado $D[v,w]$ é o conjunto de todos os vértices que pertencem a um geodésico entre v e w. Dado um conjunto S, $D[S] = \bigcup_{u,v \in S} D[u,v]$. Se $D[S] = S$, então S é um conjunto convexo. O fecho convexo de S, denotado por $H[S]$, é o menor conjunto convexo que contém S. Além disso, definimos $D_{v,w} = D[v,w] \setminus \{v,w\}$.

3.2 Problema de Classificação Geodésica

O problema de Classificação Geodésica de Grupo Único com 2 Classes (CGGU-2) foi definido em Araújo *et al.* (2019a). Nesse problema, são dados dois subconjuntos não-vazios V_B,$V_R \subseteq V$, onde $V_B \cap V_R = \varnothing$, tal que $V_{BR} = V_B \cup V_R$ representa o conjunto de amostras inicialmente classificadas e $V_N = V \setminus V_{BR}$ representa o conjunto de amostras inicialmente não classificadas. Os conjuntos V_B e V_R definem as classes azul e vermelha, respectivamente. Um exemplo de *grafo de similaridade* para o conjunto de amostras é mostrado na Figura 3.

De forma análoga à versão Euclidiana do problema, foi definido em Araújo *et al.*

Figura 3 – Exemplo de grafo de similaridade com: V_R = vértices quadrados; V_B = vértices circulares preenchidos; e V_N = vértices circulares não preenchidos.



Fonte: Araújo *et al.* (2019a)

(2019a) que G é linearmente separável em relação a $A_B \subseteq V_B$ e $A_R \subseteq V_R$ se satisfizer as três restrições abaixo, e linearmente inseparável, caso contrário.

$$H[A_B] \cap A_R = \varnothing \tag{3.1}$$

$$H[A_R] \cap A_B = \varnothing \tag{3.2}$$

$$H[A_B] \cap H[A_R] \cap V_N = \varnothing \tag{3.3}$$

As restrições (3.1) e (3.2), chamadas de restrições de separação para V_{BR}, afirmam que se um vértice inicialmente classificado está no fecho convexo da classe oposta, então o mesmo deve ser considerado como *outlier*, ou seja, não deve pertencer a A_B ou A_R. A restrição (3.3), chamada de restrição de separação para V_N, diz que não se pode ter um vértice não classificado pertencendo ao fecho convexo das duas classes ao mesmo tempo, pois não saberíamos como classificá-lo.

Se G for linearmente inseparável em relação a algum conjunto de amostras, pode se tornar linearmente separável se alguns vértices forem desconsiderados, ou seja, forem considerados como *outliers*. Neste caso, os vértices considerados *outliers* não desaparecem do grafo, o que acontece é que suas classes não são consideradas para a determinação do padrão de convexidade geodésica. Esta situação acontece no exemplo da figura 3. Nesse exemplo, G é linearmente inseparável, pois $v \in H[V_B] \cap H[V_R] \cap V_N$, mas ele se torna linearmente separável se v_1, v_2, v_3 ou v_4 for considerado *outlier*.

O problema de classificação associado a G, V_B e V_R deseja encontrar uma classificação, dada por dois conjuntos $A_B \subseteq V_B$ e $A_R \subseteq V_R$ de vértices não *outliers*, com o menor número

de vértices considerados como *outliers*. Esse problema é definido em Araújo *et al.* (2019a) como segue abaixo.

Problema 1. *Problema de Classificação Geodésica de Grupo Único com 2 Classes* (CGGU-2):

Dados G, V_B e $V_R \subseteq V$, desejamos encontrar subconjuntos para cada classe, $A_B \subseteq V_B$ e $A_R \subseteq V_R$, que definem os vértices que não são *outliers* em cada uma delas, tais que G é linearmente separável em relação a A_B e A_R, e a quantidade de *outliers* ($|V_{BR} \setminus (A_B \cup A_R)|$) seja mínima.

É válido salientar que A_B e A_R determinam um mapeamento de V nas classes $\{azul, vermelha\}$ que classifica todos os vértices não classificados em $H[A_B]$ em classe azul e todos os vértices não classificados em $H[A_R]$ em classe vermelha. Com isso, vértices não classificados podem não estar no fecho de nenhuma classe. Neste caso, podemos classificar esses vértices arbitrariamente em $V \setminus (H[A_B] \cup H[A_R])$. O tratamento desses vértices não fazem parte do escopo deste trabalho. Sendo assim, escolhemos o seguinte critério para a determinação da classe de tais vértices por questão de simplicidade e conveniência: a classe com maior quantidade de vizinhos do vértice analisado.

Note que os vértices em $(H[A_B] \cap V_R) \cup (H[A_R] \cap V_B)$ são alguns dos vértices considerados *outliers* e, mesmo sendo alcançados pelo fecho da classe oposta, não são reclassificados, ou seja, movidos para a classe oposta.

3.3 Algoritmos Probabilísticos

Em Erickson (2019), um algoritmo é definido como uma sequência de instruções para resolver um problema. Computadores são extremamente habilidosos para lidar com algoritmos, pois partindo de um estado inicial e seguindo um passo a passo bem definido, a resposta buscada será eventualmente mostrada.

Já os algoritmos probabilísticos são aqueles que utilizam experimentos aleatórios como parte de sua lógica. Na prática, isso significa que a máquina que implementa o algoritmo deve acessar um gerador de números pseudo-aleatórios. O algoritmo utiliza bits aleatórios como um guia para o seu comportamento. Diferente dos algoritmos convencionais, um algoritmo probabilístico, dada uma mesma sequência de entrada, não necessariamente leva a um mesmo estado final (CORMEN *et al.*, 2001).

Geradores de números aleatórios, por sua vez, são algoritmos que utilizam funções de dispersão e valores obtidos do relógio interno da maquina, são capazes de simular o "sorteio" de um número, tirado de um conjunto finito deles, com distribuição de probabilidade tão próxima da uniforme quanto melhor o gerador.

Recentemente, algoritmos probabilísticos têm encontrado aplicação em áreas tão distintas como computação algébrica, geometria computacional, criptografia, protocolos de redes, computação distribuída, estruturas de dados, teoria dos grafos e outras. Tudo isso motivado pelo fato de que os algoritmos probabilísticos podem ser mais simples e/ou mais eficientes quando comparados a seus correspondentes determinísticos.

Como é discutido em Figueiredo *et al.* (2007), existem dois grandes grupos de algoritmos probabilísticos, que se distinguem um do outro pela localização da incerteza que resulta da presença de experimentos aleatórios. Se a incerteza estiver na própria corretude da resposta apresentada, são ditos algoritmos de Monte Carlo, e se a incerteza estiver em seu tempo de execução, são chamados algoritmos de Las Vegas.

Dessa forma, se é preciso a garantia de que o tempo exigido pelo algoritmo será deterministicamente limitado, em todas as suas execuções, por uma certa função do tamanho da entrada, um algoritmo de Monte Carlo é certamente o mais indicado. Este tipo de algoritmo probabilístico nem sempre dá a resposta correta. Existe uma pequena chance de que a resposta apresentada esteja errada. Mas isso não é motivo para desacreditar da eficiência do algoritmo, pois a vida prática é também cheia de incerteza. Por exemplo, exames laboratoriais podem diagnosticar doenças inexistentes, mas não deixam de ser ferramentas valiosas nas mãos do bom médico. É preciso apenas saber lidar com a possibilidade de erro.

Os algoritmos probabilísticos não estão limitados a problemas de decisão. Mas para estes problemas, os algoritmos de Monte Carlo são divididos em dois tipos: os de erro unilateral e os de erro bilateral. Problemas de decisão são aqueles em que se deseja descobrir se algo é verdadeiro, a resposta certa é apenas um simples SIM ou NÃO.

Algoritmos de Monte Carlo de erro unilateral para problemas de decisão erram somente para um dos lados: aqueles que são *baseados-no-sim* nunca erram quando a resposta encontrada por eles é SIM, já os *baseados-no-não* sempre acertam quando apresentam o NÃO como resposta. Algoritmos de Monte Carlo de erro bilateral podem retornar tanto SIM quanto NÃO incorretos. A unilateralidade do erro de um algoritmo de Monte Carlo é uma característica importante que permite refinar eficientemente o nosso "grau de confiança" na resposta obtida a

níveis tão bons quanto pretendemos.

Quando se é necessário sempre ter a resposta correta, então o mais indicado é o uso de um algoritmo de Las Vegas. Este tipo de algoritmo probabilístico, além de apresentar sempre a resposta correta, tem, em geral, tempo de execução bom o suficiente para que seja justificada sua utilização, se é que apenas o aspecto simplicidade não a justificaria, e não fica devendo nada a algoritmos determinísticos quanto à qualidade de sua resposta, pois também está sempre correta. O tempo computacional de um algoritmo de Las Vegas é uma variável aleatória, cujo comportamento a análise do algoritmo torna muito bem conhecido. O tempo computacional de um algoritmo de Las Vegas pode ser e é avaliado em termos de seu valor esperado, variância ou desvio padrão.

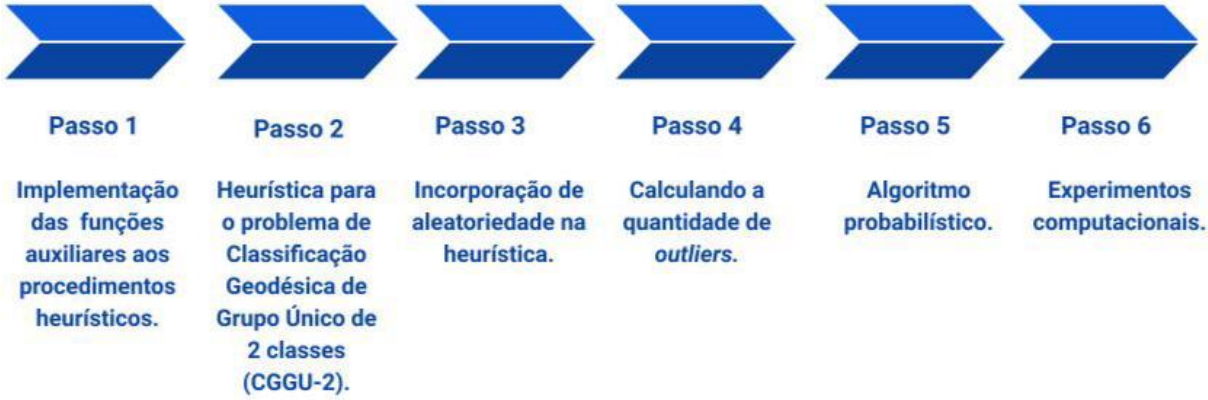
Com tudo, como foi visto, a certeza da resposta correta dada por um algoritmo de Las Vegas pode torná-lo bastante atraente. Mas o que acontece é que mesmo em se tratando de algoritmos de Las Vegas cujo tempo esperado é bom, não podemos saber ao certo se uma determinada execução do algoritmo demandará tempo muito maior. Portanto, algoritmos de Monte Carlo, por outro lado, permitem que calculemos deterministicamente seu tempo assintótico de pior caso, o que pode ser, em muitos casos, essencial.

Neste trabalho, desenvolvemos uma heurística probabilística para o problema CGGU-2, que se trata de um problema de otimização combinatória e não um problema de decisão. Como lidamos com uma heurística, não podemos garantir que nosso algoritmo probabilístico retorne sempre a melhor solução. Porém, ele irá retornar sempre uma solução viável. Dessa forma, fazendo uma analogia com as definições de tipos de algoritmos probabilísticos para os problemas de decisão descritas acima, nossa heurística se encaixaria em uma espécie de algoritmo de Monte Carlo unilateral para problema de otimização.

4 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo são descritas todas as fases necessárias para a execução deste trabalho. A Figura 4 demonstra, em sequência, os passos da execução do trabalho proposto.

Figura 4 – Diagrama exibindo o fluxo das atividades.



Fonte: Elaborada pelo autor.

4.1 Implementação das funções auxiliares aos procedimentos heurísticos

Nesta etapa apresentamos as funções auxiliares aos procedimentos heurísticos.

4.1.1 Caracterização de uma Solução

É definida a seguinte variável para representar uma solução do problema, para fins de implementação do algoritmo proposto neste trabalho:

$$a_i \in \{0,1\}, i \in V_{BR}. \tag{4.1}$$

A variável na Equação 4.1 indica quais vértices pertencem ao grupo da classe azul ou vermelha de tal modo que o grupo da classe azul seja representado por $A_B = \{i \in V_B : a_i = 1\}$ e o grupo da classe vermelha seja representado por $A_R = \{i \in V_R : a_i = 1\}$. Dessa forma, um vértice $i \in V_{BR}$ é *outlier* para uma solução com variáveis a se, e somente se, $a_i = 0$.

4.1.2 Solução Trivial

Uma solução viável para o problema CGGU-2 pode ser obtida trivialmente da seguinte forma: se $| V_B | < | V_R |$, então definimos todos os vértices em V_B como *outliers*. Caso

contrário, definimos todos os vértices em V_R como *outliers*. Com isso, é gerada uma solução viável para o problema CGGU-2 cujo valor da função objetivo é igual a $min\{\mid V_B \mid, \mid V_R \mid\}$.

4.1.3 Solução parcial inicial

Uma solução inicial para o problema CGGU-2, mas não viável, pode ser obtida da seguinte forma: define-se que cada vértice inicialmente classificado pertence ao grupo de sua classe. Consequentemente, nenhum vértice em V_{BR} é considerado como *outlier*.

4.2 Heurística para o problema CGGU-2

Nesta seção, propomos a heurística probabilística para o problema CGGU-2.

4.2.1 Configuração inicial da heurística probabilística

Inicialmente, configuramos a solução parcial inicial para o começo do desenvolvimento da solução viável retornada pela heurística.

4.2.2 Calculando o fecho convexo geodésico

Em nosso algoritmo, procuramos pelas possíveis restrições de separação linear (3.1), (3.2) e (3.3) violadas a fim de satisfazê-las. Então, precisamos calcular o fecho convexo do grupo de cada classe e verificar se alguma das restrições do problema foi violada. Para calcular o fecho convexo S de A_B (ou A_R), começamos com $S' = A_B$ (ou $S' = A_R$) e o atualizamos iterativamente. Em cada iteração, adicionamos a S' todos os vértices em D_{uv}, para cada par $u, v \in S'$ (com pelo menos um deles adicionado a S' na iteração anterior). Esta etapa é repetida até que S' não mude. Neste ponto, obtemos $S = S'$. Os conjuntos D_{uv} podem ser determinados a priori com um algoritmo semelhante ao *Breadth First Search* (BFS).

Usamos uma variável z para armazenar informações do calculo do fecho convexo de um grupo. A mesma é definida na Equação 4.2 da seguinte forma:

$$z_{ki} \in \{0,1\}, k \in \{B,R\}, i \in V. \tag{4.2}$$

A variável z indica quais vértices pertencem ao fecho convexo do grupo da classe k (azul ou vermelha), onde $k \in \{B,R\}$, de tal forma que o fecho convexo do grupo k é representado por $S = \{i \in V : z_{ki} = 1\}$.

4.2.3 Restrições de separação para V_{BR}

Se uma das restrições de separação para V_{BR} (3.1) ou (3.2) for violada, significa que existe algum vértice $i \in V_{BR}$ pertencente ao fecho convexo do grupo de classe oposta. Neste caso, retiramos i do grupo em que estava, considerando-o como *outlier*. Em seguida, recalculamos o fecho convexo do grupo ao qual i pertencia.

4.2.4 Restrição de separação para V_N

Se a restrição de separação para V_N (3.3) for violada, então existe algum vértice $i \in V_N$ pertencente ao fecho convexo do grupo azul e ao fecho convexo do grupo vermelho. Neste caso, consideramos um vértice $i \in A_B \cup A_R$ como *outlier*, retirando-o do grupo ao qual pertence. A escolha de tal vértice i é dada por uma da função que define o critério de escolha. Em seguida, recalculamos o fecho convexo do grupo ao qual i pertencia.

4.3 Incorporação de aleatoriedade na heurística

Definimos uma função, a partir de um certo critério, que determina um vértice i como *outlier*, a chamamos de *setOutlier*. A mesma será implementada de seis formas diferentes, cada uma usando um critério de escolha diferente. Depois de determinar o vértice que irá se tornar *outlier*, atualiza-se o fecho convexo do grupo no qual ele pertencia. Os critérios são descritos a seguir.

4.3.1 Critério 1: C1-Me

A primeira forma de ser implementada consiste em escolher aleatoriamente um vértice i dentro do grupo de menor cardinalidade para torná-lo *outlier*.

4.3.2 Critério 2: C2-Ma

A segunda forma de ser implementada consiste em escolher aleatoriamente um vértice i dentro do grupo de maior cardinalidade para torná-lo *outlier*.

4.3.3 Critério 3: C3-MeMa

A terceira forma de ser implementada consiste em escolher aleatoriamente um vértice *i* dentro do grupo de menor cardinalidade dando prioridade, através de pesos nos vértices, para o vértice de maior grau. Definimos os pesos dos vértices como o grau de cada vértice. Para isso, criamos uma função para calcular o grau (quantidade de vizinhos) de cada vértice e uma função de probabilidade para escolher um vértice para ser retirado do grupo (se tornar *outlier*). Cada vértice *i* pertencente ao grupo considerado terá $grau[i])/somaGraus$ de chance de ser escolhido, onde $grau[i]$ é o grau de *i* no grafo *G* e *somaGraus* é a soma dos graus de todos os vértices desse grupo.

4.3.4 Critério 4: C4-MeMe

A quarta forma de ser implementada consiste em escolher aleatoriamente um vértice *i* dentro do grupo de menor cardinalidade dando prioridade, através de pesos nos vértices, para o vértice de menor grau. O cálculo do peso de um vértice *i* é dado pela divisão do MMC dos graus dos vértices do grupo considerado pelo grau de *i* em *G* ($peso[i] = mmc/grau[i]$, onde *i* é um vértice do grupo). Utilizamos as mesmas funções comentadas no critério C3-MeMa para calcular o grau dos vértices e para escolher o vértice a ser retirado do grupo, com a única diferença sendo os pesos dos vértices.

4.3.5 Critério 5: C5-MaMa

A quinta forma de ser implementada consiste em escolher aleatoriamente um vértice *i* dentro do grupo de maior cardinalidade dando prioridade, através de pesos nos vértices, para o vértice de maior grau. Da mesma forma apresentada no critério C3-MeMa, definimos os pesos dos vértices como o grau de cada vértice e utilizamos as mesmas funções para calcular o grau dos vértices e para escolher o vértice a ser retirado do grupo.

4.3.6 Critério 6: C6-MaMe

A sexta forma de ser implementada consiste em escolher aleatoriamente um vértice *i* dentro do grupo de maior cardinalidade dando prioridade, através de pesos nos vértices, para os vértices de menor grau. Para encontrar os pesos dos vértices utilizamos o mesmo cálculo explicado no critério C4-MeMe, que é dado pela divisão do MMC dos graus dos vértices do

grupo considerado pelo grau do vértice. Por fim, fizemos uso das mesmas funções explicadas no critério C3-MeMa para calcular o grau dos vértices e para escolher o vértice a ser retirado do grupo, com a diferença sendo os pesos dos vértices.

4.3.7 Resumo dos Critérios

A Tabela 2 mostra um resumo da diferença entre os critérios. A mesma pode ser lida, por exemplo, da seguinte forma: O critério C3-MeMa escolhe aleatoriamente um vértice *i* para tornar *outlier* no grupo de cardinalidade menor, priorizando o vértice de grau maior. Com isso, podemos notar que a parte "Me" do nome critério vem da escolha do vértice no grupo de menor cardinalidade, e a parte "Ma" vem da priorização do vértice de maior grau.

Tabela 2 – Tabela mostrando a diferença entre os critérios.

	Escolhe aleatoriamente um vértice para se tornar *outlier* no grupo de	
	Cardinalidade	Priorizando o vértice de grau
C1-Me	menor	-
C2-Ma	maior	-
C3-MeMa	menor	maior
C4-MeMe	menor	menor
C5-MaMa	maior	maior
C6-MaMe	maior	menor

Fonte: Elaborada pelo autor.

4.4 Calculando a quantidade de *outliers*

Depois que todas as restrições de separação (3.1), (3.2) e (3.3) do problema são satisfeitas, calculamos a quantidade de vértices que se tornaram *outliers*, cuja minimização é o critério de otimização do problema. Para esse cálculo, consideramos que para ser *outlier* o vértice precisa ser inicialmente classificado e não pertencer ao grupo da sua classe. É importante lembrar que as restrições de separação para V_{BR} (3.1) e (3.2) dizem que um vértice inicialmente classificado não deve pertencer ao fecho convexo do grupo da classe oposta, mas se pertencer, o mesmo deve ser considerado como *outlier* para a solução do nosso problema. E a restrição de separação para V_N (3.3) diz que um vértice inicialmente não classificado não pode pertencer ao fecho convexo do grupo da classe azul e ao fecho convexo do grupo da classe vermelha simultaneamente, pois não saberíamos como classificá-lo.

4.5 Algoritmo Probabilístico Final

O pseudocódigo do algoritmo probabilístico geral que representa a heurística probabilística para o problema CGGU-2 funciona da seguinte forma: inicialmente, é configurada a solução parcial inicial e calculado o fecho convexo dos dois grupos. Logo depois, para todo vértice i em V, o algoritmo verifica se alguma restrição de separação linear é violada. Se a restrição violada for a restrição de separação para V_{BR} (3.1) ou (3.2), i é definido como *outlier*. Já se a restrição violada for a restrição de separação para V_N (3.3), é chamada a função *setOutlier* para escolher, baseada em critérios, o vértice que vai se tornar outlier. Com isso, depois que todas as restrições forem satisfeitas, é calculada a quantidade de outliers e feito um teste simples, onde é verificado se a quantidade de *outliers* gerados é maior do que a quantidade de vértices do menor conjunto: $min\{\mid V_B \mid, \mid V_R \mid\}$. Caso seja confirmado que a quantidade de *outliers* é maior do que a cardinalidade do menor conjunto, a solução retornada é a trivial, em que o menor conjunto é atualizado como um conjunto composto completamente de *outliers*. O pseudocódigo do algoritmo probabilístico é apresentado no Algoritmo 1.

Algoritmo 1: Algoritmo probabilístico final da heurística probabilística

```
Configurar a solução parcial inicial;
Calcular o fecho convexo dos dois grupos;
for cada i ∈ V do
    if i ∈ A_B ∩ H[A_R] ou i ∈ A_R ∩ H[A_B] then
        Definir o vértice i como outlier;
    end
    if i ∈ V_N ∩ H[A_B] ∩ H[A_R] then
        setOutlier();
    end
end
Calcular a quantidade de outliers;
if quantidade de outliers é menor que min{| V_B |, | V_R |} then
    return a quantidade de outliers;
else
    return a solução trivial;
end
```

5 EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

Neste capítulo, apresentaremos a forma que os testes foram realizados e os resultados obtidos.

5.1 Ambiente de testes

Os testes foram realizados em um notebook com a seguinte configuração: Processador AMD Ryzen 7 Quad Core 2.30 GHz até 4.00 GHz, memória RAM de 8 GB DDR4 2666 MHz, SSD de 128 GB e sistema operacional Ubuntu 20.04 LTS.

5.2 Instâncias Sintéticas e Instâncias Realistas

As instâncias sintéticas e instâncias realistas foram derivadas de instâncias (conjunto de dados) da versão Euclidiana do problema de classificação. A partir de cada conjunto de dados, foram criadas 20 instâncias associadas à versão geodésica do problema de classificação da seguinte maneira: construímos o grafo de similaridade associado usando o conceito de transformação de Zaki e Meira Jr (2014), onde cada ponto se torna um vértice do grafo. Dessas 20 instâncias, 10 possuem 80% de vértices inicialmente classificados e as outras 10 possuem 70%, escolhendo aleatoriamente 20% (ou 30%) dos vértices para se tornarem não classificados. Esses vértices formam o conjunto de validação. Essa transformação do conjunto de dados em um grafo de similaridade é descrita a seguir: cada vértice i do grafo representa um ponto x_i do conjunto de dados. Para criar as arestas, primeiro calculamos a similaridade entre os pontos, para isso foi usada a função de kernel de Gauss, dada por:

$$s_{ij} = exp\{\frac{-||x_i - x_j||^2}{2}\},$$

onde formula que $|| x_i - x_j ||^2 = \sum_{k=1}^{22} (x_i(k) - x_j(k))^2$ é o quadrado da distância Euclidiana entre os pontos x_i e x_j. Dessa forma, cada aresta (x_i, x_j) tem um peso de similaridade s_{ij} correspondente ao valor de similaridade entre x_i e x_j. Posteriormente, para cada vértice i, foi calculado o vetor q de vizinhos mais próximos em termos do valor de similaridade, dado como:

$$N_q(i) = \{j \in V : \ j \neq i, \ s_{ij} \geq s_{iq}\},$$

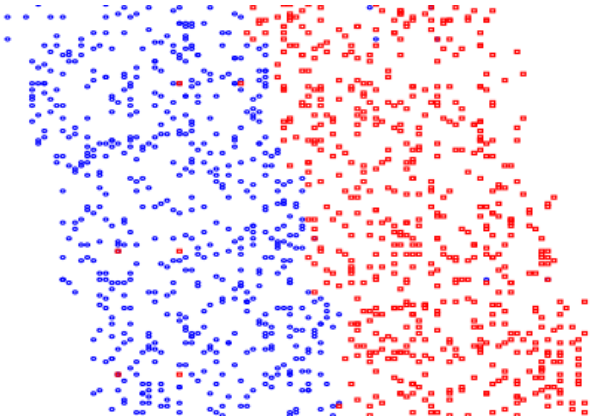
onde s_{iq} representa o valor de similaridade entre i e seu q-ésimo vizinho mais próximo (usamos q = 10 % de $|V|$). Portanto, uma aresta é adicionada entre os vértices i e j se, e somente se, ambos i e j são vizinhos mais próximos entre si, isto é, $j \in N_q(i)$ e $i \in N_q(j)$. Por fim, se o resultado for um grafo desconectado, adicionamos arestas mais altas (ou seja, com maior ponderação) entre

todos os pares de componentes conectados.

5.2.1 Instâncias Sintéticas

Uma instância sintética é uma instância gerada aleatoriamente, com um certo padrão, onde a solução ótima já é conhecida de antemão. Como em Corrêa *et al.* (2019a), geramos duas bases de instâncias sintéticas para o problema de classificação Euclidiana, como descrevemos a seguir. Dada a dimensão do espaço $d \in \{2,3\}$ e um hiperplano em $\mathbb{R}^d$, escolhemos um ponto em cada metade do espaço, digamos b e r, cada um a uma distância 1 do hiperplano. O ponto b (respectivamente r) funcionará como o centro para as amostras azuis (resp. vermelhas). Então, consideramos um hipercubo $d-$dimensional com o lado 2 centrado em cada ponto central. Geramos aleatoriamente p pontos uniformemente distribuídos dentro dele. Por fim, geramos 5 *outliers* para cada classe. Usamos $p = 50$ e $n = 5$. Um exemplo de instância base para $d = 2$ é mostrada na Figura 5.

Figura 5 – Exemplo de instância sintética com $d = 2$.



Fonte: Corrêa *et al.* (2019a)

5.2.2 Instâncias Realistas

Para testarmos os algoritmos desenvolvidos para aplicações realistas, realizamos experimentos usando instâncias derivadas de dois conjuntos de dados: doença de Parkinson (LITTLE *et al.*, 2007) e imagens para diagnósticos cardíacos *Single Proton Emission Computed Tomography* (SPECT) (KURGAN *et al.*, 2001).

5.2.2.1 Instâncias da doença de Parkinson

As instâncias da doença de Parkinson representam pessoas que podem ter ou não ter uma doença conhecida como Parkinson. O conjunto de dados é composto por 195 pontos, cada um deles guardando informações de medição de voz biomédica de uma pessoa divididas em 22 atributos. Para esses 195 pontos, 147 desses indicam pessoas com doença de Parkinson e 48 indicam pessoas saudáveis.

5.2.2.2 Instâncias de imagens cardíacas SPECT

O conjunto de dados cardíaco descreve o diagnóstico de doenças pelas imagens cardíacas geradas pelo SPECT. Cada um dos pacientes é classificado em duas categorias (classes): normais e anormais. O banco de dados consiste em 267 exames SPECT de pacientes que foram processados para extrair recursos que resumem as imagens SPECT originais. Como resultado, 44 padrões de recursos contínuos foram criados para cada paciente, além do estado de diagnóstico (normal ou anormal).

5.3 Algoritmos da versão Euclidiana

Nesta seção, descrevemos brevemente sobre os algoritmos da versão Euclidiana do problema usados para fins de comparação em nossos experimentos.

5.3.1 Algoritmo SVM

Um *Support Vector Machine* (SVM) é um um conjunto de métodos de aprendizado supervisionado que analisam os dados, reconhecem padrões e classificam os dados. É formalmente definido por um hiperplano de separação. Em outras palavras, dados os dados de treinamento rotulados (aprendizado supervisionado), o algoritmo produz um hiperplano ótimo que classifica novas amostras. Em um espaço bidimensional, esse hiperplano é uma linha que divide um plano em duas partes, onde cada classe fica em um dos lados.

Em nossos experimentos, usamos a implementação do algoritmo *linearSVC* do SVM (disponível em: https://scikit-learn.org/stable/modules/svm.html) como um algoritmo de classificação Euclidiana. Ele usa um *kernel* linear, o que significa que ele tenta separar o conjunto de dados linearmente. Dessa forma, é apropriado para comparação com nosso algoritmo de

classificação geodésica, uma vez que o mesmo foi desenvolvido para a versão de grupo único.

5.3.2 Algoritmo de Redes Neurais MLP

Como outro algoritmo de classificação Euclidiana, usamos uma implementação do algoritmo de redes neurais *Multi-Layer Perceptron* (MLP), que aplica *backpropagation* (retropropagação). Para mais detalhes, consulte https://scikit-learn.org/stable/modules/neural_networks_supervised.html.

5.4 Resultados constatados

Para cada grupo de instâncias, executamos cada instância 10 vezes e tiramos a média do valor da solução, da acurácia e do tempo de execução. Logo depois, foi calculado a média das médias das instâncias e gerado os valores para cada grupo de instâncias, que são apresentados nas Tabelas 4, 5, 6, 7 e 8.

As informações a seguir são importantes para o entendimento das tabelas:

- Nome da instância: nome da instância usando o formato: nome da instância, número de vértices em $G\,(v)$, densidade de $G\,(d)$ e porcentagem de vértices inicialmente classificados (br);
- v: número de vértices do grafo
- m: número de arestas do grafo
- Diam: diâmetro da instância do grafo;
- Dgmin: grau mínimo da instância do grafo;
- Dgmax: grau máximo da instância do grafo;
- OPT: solução ótima (número mínimo de *outliers*) para as instâncias (encontrada em Araújo *et al.* (2019b));
- CG: acurácia das soluções do algoritmo exato encontrada em Araújo *et al.* (2019b);
- SVM: acurácia das soluções do algoritmo SVM;
- MLP: acurácia das soluções do algoritmo MLP.

A Tabela 3 apresenta os tipos de instâncias utilizadas nos experimentos e suas propriedades (ARAÚJO *et al.*, 2019b).

A Tabela 4 apresenta a comparação do valor da solução entre os seis critérios do algoritmo probabilístico e a solução ótima encontrada no algoritmo exato CG desenvolvido em

Tabela 3 – Tabela mostrando as propriedades de cada tipo de instância.

Nome da instância	v	m	Dens	Diam	Dgmin	Dgmax
syntheticDim2	104	384	7	17	1	9
syntheticDim3	96	315	6	10	2	8
parkinsons	195	1097	5	10	1	18
spectf	267	1826	5	8	1	36

Fonte: Elaborada pelo autor.

Araújo *et al.* (2019b).

Tabela 4 – Tabela comparando o valor da solução entre os seis critérios do algoritmo probabilístico e o valor ótimo.

Nome da instância	Valor da solução (número de *outliers*)						
	OPT	C1-Me	C2-Ma	C3-MeMa	C4-MeMe	C5-MaMa	C6-MaMe
syntheticDim2_v104_d7_br70	11,90	**34,40**	34,70	34,49	34,65	34,70	34,70
syntheticDim2_v104_d7_br80	10,60	**40,30**	**40,30**	**40,30**	**40,30**	**40,30**	**40,30**
syntheticDim3_v96_d6_br70	13,00	30,50	30,60	30,45	**30,14**	30,60	30,60
syntheticDim3_v96_d6_br80	15,40	**33,80**	34,30	34,27	34,30	34,30	34,30
parkinsons_v195_d5_br70	34,10	**34,40**	**34,40**	**34,40**	**34,40**	**34,40**	**34,40**
parkinsons_v195_d5_br80	36,90	**37,50**	**37,50**	**37,50**	**37,50**	**37,50**	**37,50**
spectf_v267_d5_br70	39,20	**39,80**	**39,80**	**39,80**	**39,80**	**39,80**	**39,80**
spectf_v267_d5_br80	43,40	**44,20**	**44,20**	**44,20**	**44,20**	**44,20**	**44,20**
Média	25,56	**36,86**	36,98	36,93	36,91	36,98	36,98

Fonte: Elaborada pelo autor.

A Tabela 5 apresenta a comparação da acurácia entre os seis critérios algoritmo probabilístico e o algoritmo CG.

Tabela 5 – Tabela comparando a acurácia entre os critérios do algoritmo probabilístico e o algoritmo CG.

Nome da instância	Acurácia (%)						
	C1-Me	C2-Ma	C3-MeMa	C4-MeMe	C5-MaMa	C6-MaMe	CG
syntheticDim2_v104_d7_br70	50,64	**93,22**	50,35	50,58	71,29	91,26	92,90
syntheticDim2_v104_d7_br80	53,50	76,50	50,58	51,17	63,95	85,65	**88,50**
syntheticDim3_v96_d6_br70	70,36	**84,29**	70,36	70,96	83,07	**89,72**	83,21
syntheticDim3_v96_d6_br80	74,74	75,26	76,21	74,74	88,10	84,55	**88,42**
parkinsons_v195_d5_br70	**76,73**	68,41	**76,73**	**76,73**	67,52	69,24	76,72
parkinsons_v195_d5_br80	**73,08**	67,97	**73,08**	**73,08**	71,41	60,21	**73,08**
spectf_v267_d5_br70	**81,00**	71,86	**81,00**	**81,00**	70,98	64,41	**81,00**
spectf_v267_d5_br80	**79,62**	68,26	**79,62**	**79,62**	69,13	65,04	**79,62**
Média	69,96	75,72	69,74	69,74	73,18	76,26	**82,93**

Fonte: Elaborada pelo autor.

A Tabela 6 apresenta a comparação da acurácia entre os seis critérios do algoritmo probabilístico e o algoritmo SVM.

Tabela 6 – Tabela comparando a acurácia entre os critérios do algoritmo probabilístico e o algoritmo SVM.

Nome da instância	Acurácia (%)						
	C1-Me	C2-Ma	C3-MeMa	C4-MeMe	C5-MaMa	C6-MaMe	SVM
syntheticDim2_v104_d7_br70	50,64	**93,22**	50,35	50,58	71,29	**91,26**	85,16
syntheticDim2_v104_d7_br80	53,50	76,50	50,58	51,17	63,95	**85,65**	79,00
syntheticDim3_v96_d6_br70	70,36	84,29	70,36	70,96	83,07	89,72	**94,64**
syntheticDim3_v96_d6_br80	74,74	75,26	76,21	74,74	88,10	84,55	**94,74**
parkinsons_v195_d5_br70	**76,73**	68,41	**76,73**	**76,73**	67,52	69,24	65,17
parkinsons_v195_d5_br80	**73,08**	67,97	**73,08**	**73,08**	71,41	60,21	68,72
spectf_v267_d5_br70	**81,00**	71,86	**81,00**	**81,00**	70,98	64,41	66,75
spectf_v267_d5_br80	**79,62**	68,26	**79,62**	**79,62**	69,13	65,04	73,58
Média	69,96	75,72	69,74	69,74	73,18	76,26	**78,47**

Fonte: Elaborada pelo autor.

A Tabela 7 apresenta a comparação da acurácia entre os seis critérios do algoritmo probabilístico e o algoritmo MLP.

Tabela 7 – Tabela comparando a acurácia entre os critérios do algoritmo probabilístico e o algoritmo MLP.

Nome da instância	Acurácia (%)						
	C1-Me	C2-Ma	C3-MeMa	C4-MeMe	C5-MaMa	C6-MaMe	MLP
syntheticDim2_v104_d7_br70	**50,64**	93,22	**50,35**	**50,58**	71,29	91,26	44,19
syntheticDim2_v104_d7_br80	**53,50**	76,50	**50,58**	**51,17**	63,95	85,65	41,50
syntheticDim3_v96_d6_br70	**70,36**	84,29	**70,36**	**70,96**	83,07	89,72	48,57
syntheticDim3_v96_d6_br80	**74,74**	75,26	**76,21**	**74,74**	88,10	84,55	48,95
parkinsons_v195_d5_br70	**76,73**	68,41	**76,73**	**76,73**	67,52	69,24	76,72
parkinsons_v195_d5_br80	73,08	67,97	73,08	73,08	71,41	60,21	**75,38**
spectf_v267_d5_br70	**81,00**	71,86	**81,00**	**81,00**	70,98	64,41	80,00
spectf_v267_d5_br80	**79,62**	68,26	**79,62**	**79,62**	69,13	65,04	79,25
Média	69,96	75,72	69,74	69,74	73,18	76,26	**61,82**

Fonte: Elaborada pelo autor.

A Tabela 8 apresenta a comparação do tempo de execução entre os seis critérios do algoritmo probabilístico.

Tabela 8 – Tabela comparando o tempo de execução entre os seis critérios do algoritmo probabilístico.

Nome da instância	Tempo de execução (s)					
	C1-Me	C2-Ma	C3-MeMa	C4-MeMe	C5-MaMa	C6-MaMe
syntheticDim2_v104_d7_br70	**0,04**	0,06	**0,04**	0,10	0,07	0,06
syntheticDim2_v104_d7_br80	**0,05**	0,08	**0,05**	0,08	0,12	0,07
syntheticDim3_v96_d6_br70	**0,03**	0,06	0,05	**0,03**	0,05	0,06
syntheticDim3_v96_d6_br80	0,04	**0,03**	0,07	0,04	0,06	0,06
parkinsons_v195_d5_br70	**0,25**	0,66	0,40	0,28	0,56	0,92
parkinsons_v195_d5_br80	**0,26**	0,70	0,51	0,28	0,75	1,02
spectf_v267_d5_br70	3,26	6,01	**2,88**	8,98	5,49	23,27
spectf_v267_d5_br80	**4,21**	7,20	4,54	7,14	7,40	20,06
Média	**1,02**	1,85	1,07	2,12	1,81	5,69

Fonte: Elaborada pelo autor.

5.4.1 Valor da Solução

Na comparação por valor da solução, analisamos a variação do valor da solução (número de *outliers*) dos critérios com a solução ótima (OPT). Para as instâncias sintéticas, nenhum critério conseguiu chegar próximo ao valor da solução ótima. Já para as instâncias realistas, os valores chegaram próximo ao ótimo, mas não chegaram a ser iguais.

A média do critério C1-Me (critério que teve a melhor média entre os seis critérios) foi 36,86 e a média da solução ótima foi 25,56. Observa-se que os valores entre os critérios não têm diferença significativa. Porém, por se tratar de uma heurística e o valor de solução estar longe do valor ótimo, ainda assim os resultados evidenciam uma diferença significativa na acurácia entre os critérios, como pode ser visto na subseção seguinte. Isso sugere que a função objetiva do problema, que minimiza a quantidade de *outliers*, não seja um bom critério para atingirmos uma boa acurácia, pelo menos ao usarmos heurísticas.

É importante notar que as instâncias realistas são instâncias cujo valor ótimo é muito alto devido à limitação da versão de grupo único do problema, o que impossibilita a identificação de uma maior variedade de padrões das amostras. Já as instâncias sintéticas foram criadas para exemplificar um grupo de instâncias cuja quantidade de *outliers* na solução ótima é sempre baixa.

5.4.2 Acurácia

Analisamos a acurácia da classificação dos vértices inicialmente não classificados para os seis critérios da heurística probabilística descrita no Algoritmo 1 e comparamos com a acurácia dos algoritmos MLP, SVM e CG.

Em média, os resultados de todos os critérios foram melhores que os resultados do algoritmo MLP na média geral. Mas, os melhores critérios foram os critérios C1-Me, C3-MeMa e C4-MeMe que se destacaram por terem obtido acurácia melhor que a do algoritmo MLP em 87% dos casos.

Para as instâncias sintéticas, apenas os critérios C2-Ma e C6-MaMe conseguiram acurácia melhor do que a do algoritmo SVM, que funciona bem para tais instâncias. O critério C2-Ma foi melhor em 25% dos casos e o critério C6-MaMe foi melhor em 50% dos casos. Para as instâncias de Parkinson, os melhores resultados foram os dos critérios C1-Me, C3-MeMa e C4-MeMe, pois foram melhores que os resultados dos outros critérios e do algoritmo SVM em todos os casos. E, para as instâncias SPECT, em 67% dos grupos de instâncias foram melhores

do que os resultados do algoritmo SVM. Para ser mais específico, os resultados dos critérios C1-Me, C3-MeMa e C4-MeMe foram melhores que os resultados do algoritmo SVM em todos os casos.

Portanto, para a comparação com o algoritmo SVM, podemos concluir que o critério C6-MaMe se destacou para as instâncias sintéticas, pois conseguiu resultados melhores que os resultados do algoritmo SVM em 50% dos casos. E os critérios C1-Me, C3-MeMa e C4-MeMe se destacaram por conseguirem resultados melhores na comparação com o algoritmo SVM para as instâncias de Parkinson e SPECT.

Para as instâncias sintéticas, em 25% dos casos os critérios 2 e 6 foram melhores que os resultados do algoritmo CG. É importante ressaltar que o critério C6-MaMe se destacou por te obtido resultados muito próximos aos resultados do algoritmo CG. Em média, conseguiu 99% da acurácia que o algoritmo CG obteve em relação às instâncias sintéticas. Já para as instâncias de Parkinson e SPECT, os critérios C1-Me, C3-MeMa e C4-MeMe tiveram resultados iguais ou melhores que os resultados do algoritmo CG em 75% dos casos.

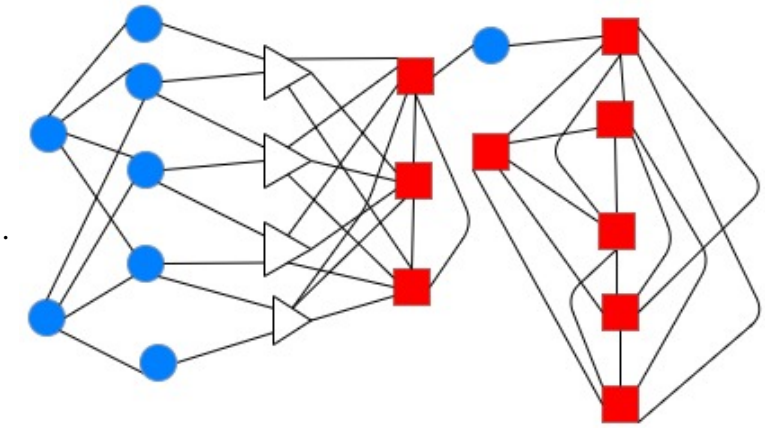
No geral, concluímos que para as instâncias sintéticas, o critério C6-MaMe foi o melhor entre os seis critérios, pois conseguiu os melhores resultados. Nesse caso, vale ressaltar que o melhor vértice a ser escolhido para se tornar *outlier* é o vértice que se encontra no grupo de maior cardinalidade, e que vale a pena utilizar a função de probabilidade que dá prioridade ao vértice de menor grau. Além disso, ainda para as instâncias sintéticas, o critério C6-MaMe foi totalmente melhor que o algoritmo MLP, melhor que o algoritmo SVM em 50% dos casos e melhor que o algoritmo CG em apenas 25% dos casos.

Já para as instâncias realistas de Parkinson e SPECT, os critérios C1-Me, C3-MeMa e C4-MeMe conseguiram os melhores resultados. Mas, entre eles, escolhemos o critério C1-Me como o melhor critério pela simplicidade. Nesse caso, é válido salientar que o melhor vértice a ser escolhido para se tornar *outlier* é o vértice escolhido aleatoriamente no grupo de menor cardinalidade, que é um critério bem diferente do critério C6-MaMe. Ainda para as instâncias de Parkinson e SPECT, o critério C1-Me foi melhor que o algoritmo MLP em 75% dos casos, totalmente melhor que o algoritmo SVM e melhor ou igual ao algoritmo CG em todos os casos.

Vale salientar que, apesar do algoritmo CG ser um algoritmo exato, a acurácia dele não é 100% porque a estratégia de identificação de padrão (função objetivo e restrições que definem o problema) utilizada não consegue identificar todos os padrões possíveis de existir. Assim como nenhuma técnica consegue prever todos os possíveis padrões que existem no mundo.

Para as instâncias de nossos experimentos, observamos que grande parte dos bons resultados das heurísticas probabilísticas que propomos vêm da parte da classificação dos vértices não alcançados por nenhum fecho convexo na solução retornada. Como mencionado na Seção 3.2, definimos a classe de tais vértices a partir da sua vizinhança, pois essa questão não faz parte do escopo deste trabalho. Apesar da escolha desse padrão não ser elaborada, bons resultados aparecem para muitas das instâncias que testamos. Porém, há instâncias em que tal estratégia de definição de classe para esses vértices não alcançados por nenhum fecho convexo não funciona bem. Um exemplo disso pode ser visto na Figura 6, onde a definição de classe para os vértices não classificados seria totalmente errada, com 0% de acurácia, ao usar apenas o critério de vizinhança, pois os vértices inicialmente não classificados (vértices triangulares brancos) seriam classificados como vermelhos (vértices quadrados) por terem mais vizinhos vermelhos que azuis (vértices circulares), mas a verdadeira cor deles é azul para essa instância. No entanto, para todos os critérios da nossa heurística probabilística e o algoritmo exato em Araújo *et al.* (2019b) retornariam soluções com 100% de acurácia pois os vértices inicialmente não classificados estariam no fecho convexo apenas do grupo da classe azul e seriam classificados corretamente como azuis. Vale salientar que estamos tratando da versão de grupo único do problema de Classificação Geodésica e, por isso, há limitações na identificação de padrões.

Figura 6 – Exemplo de instância para o problema CGGU-2



Fonte: Elaborada pelo autor.

Por fim, podemos notar que não há uma diferença significativa em relação às prioridades dos vértices para os critérios que escolhem como *outliers* os vértices no menor grupo (critérios C1-Me, C3-MeMa e C4-MeMe). Porém, para os demais critérios, que escolhem como *outliers* os vértices no maior grupo, a diferença é significativa, com uma leve vantagem para a priorização dos vértices de menor grau (critério C6-MaMe), que na média geral de acurácia

levou ao melhor resultado, inclusive considerando todos os seis critérios.

5.4.3 Tempo de Execução

Na comparação pelo tempo de execução, analisamos a variação de tempo para cada grupo de instâncias. Podemos notar que para todos os critérios, e para as instâncias sintéticas e instâncias de Parkinson, o tempo de execução foi menor ou igual a 1 segundo. Já para as instâncias SPECT o tempo aumentou um pouco, em 83% dos resultados o tempo variou entre 3 e 7 segundos, e em 17% dos casos o tempo variou entre 20 e 23 segundos. É válido salientar que, para todas as instâncias, o tempo de execução dos algoritmos MLP e SVM é em média 0,01 segundo, e do algoritmo CG são em média, para as instâncias sintéticas, 0,52 segundos; para as instâncias de Parkinson, 15 segundos e para as instâncias SPECT, 0,21 segundos. E em média, para todas as instâncias, o tempo de execução do algoritmo CG foi 5,24 segundos, ou seja, maior que o tempo de execução do critério C1-Me, que foi 1,02 segundos. Contudo, em geral, em 75% dos resultados o tempo de execução do algoritmo foi menor ou igual a 1 segundo. Em média, o critério C1-Me conseguiu o melhor tempo com 1,02 segundos e o critério C6-MaMe obteve o pior tempo com 5,69 segundos. Portanto, por tempo de execução, podemos concluir que nosso algoritmo teve resultados condizentes com uma heurística rápida, o que ajuda a utilizarmos tais algoritmos dentro de uma outra estratégia, como por exemplo uma meta-heurística.

6 CONCLUSÕES E TRABALHOS FUTUROS

Desenvolvemos uma heurística probabilística para o problema de CGGU-2 e propomos 6 variações no critério de escolha dos vértices para se tornarem *outliers*. Nos experimentos computacionais, comparamos os resultados de nossa heurística com alguns algoritmos conhecidos na literatura (MLP, SVM e CG). Pelos experimentos, temos evidências de que o problema se mostra difícil até para a obtenção de uma simples solução viável melhor que a solução trivial, considerando o critério de otimização como sendo a minimização da quantidade de *outliers*.

É importante ressaltar que a nossa heurística probabilística não obteve valor de solução e acurácia melhores que os resultados já existentes na literatura para a versão geodésica, o que já era esperado, visto que o algoritmo que apresentou os melhores resultados é um algoritmo exato. No entanto, o tempo de execução da heurística é muito pequeno para todas as instâncias testadas, o que evidencia a viabilidade para o uso em instâncias maiores do problema ou incorporação a uma outra estratégia, como meta-heurística. No quesito acurácia, os resultados de alguns dos critérios da nossa heurística foram melhores que os resultados já conhecidos pelos algoritmos MLP e SVM. Portanto, podemos considerar como promissora a heurística probabilística ao considerarmos o uso de determinados critérios de acordo com determinados grupos de instâncias que foram testados.

Para trabalhos futuros, pretendemos fazer testes com outros tipos de instâncias para avaliar melhor os algoritmos. Além disso, planejamos desenvolver uma implementação da meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) utilizando os algoritmos propostos neste trabalho. Outra ideia é tratarmos uma versão estendida do problema CG que permite mais de um grupo por classe, cujo intuito é ter a possibilidade de identificar novos padrões de convexidade geodésica usando uma quantidade menor de *outliers*.

REFERÊNCIAS

ARAÚJO, P. H. M.; CAMPÊLO, M. B.; CORRÊA, R. C.; LABBÉ, M. The geodesic classification problem on graphs. **Latin e American Algorithms, Graphs and Optimization Symposium**, 2019.

ARAÚJO, P. H. M. de; CAMPÊLO, M.; CORRÊA, R. C. **The Geodesic Classification Problem on Graphs**. Tese (Doutorado) – PhD thesis, Mestrado e Doutorado em Ciência da Computação, Departamento de Ciência da Computação da Universidade Federal do Ceará, 2019.

ARTIGAS, D.; DANTAS, S.; DOURADO, M.; SZWARCFITER, J. Partitioning a graph into convex sets. **Discrete Mathematics**, v. 311, p. 1968–1977, 09 2011.

BERTSIMAS, D.; SHIODA, R. Classification and regression via integer optimization. **Operations Research**, INFORMS, v. 55, n. 2, p. 252–271, 2007.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms**. 2nd. ed. [*S. l.*]: The MIT Press, 2001. ISBN 0262032937.

CORRÊA, R. C.; BLAUM, M.; MORENCO, J.; KOCH, I.; MYDLARZ, M. An integer programming approach for the 2-class single-group classification problem. **Latin e American Algorithms, Graphs and Optimization Symposium**, 2019.

CORRÊA, R. C.; DONNE, D. D.; MARENCO, J. On the combinatorics of the 2-class classification problem. **Discrete Optimization**, v. 31, p. 40 – 55, 2019. ISSN 1572-5286. Disponível em: http://www.sciencedirect.com/science/article/pii/S1572528617302748.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, Springer, v. 20, p. 273–297, 1995.

ERICKSON, J. **Algorithms**. [*S. l.*: *s. n.*], 2019. ISBN 978-1-792-64483-2.

FIGUEIREDO, C.; FONSECA, G. da; LEMOS, M.; Sá, V. Pereira de. **Introdução aos Algoritmos Randomizados**. [*S. l.*]: Instituto Nacional de Matemática Pura e Aplicada, 2007. ISBN 978-85-244-0255-5.

KURGAN, L. A.; CIOS, K. J.; TADEUSIEWICZ, R.; OGIELA, M.; GOODENDAY, L. S. Knowledge discovery approach to automated cardiac spect diagnosis. **Artificial Intelligence in Medicine**, v. 23, n. 2, p. 149–169, 2001. ISSN 0933-3657. Disponível em: https://www.sciencedirect.com/science/article/pii/S0933365701000823.

LITTLE, M. A.; MCSHARRY, P. E.; ROBERTS, S. J.; COSTELLO, D. A.; MOROZ, I. M. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. **BioMedical Engineering OnLine**, v. 6, n. 1, p. 23, Jun 2007. ISSN 1572-5286. Disponível em: https://doi.org/10.1186/1475-925X-6-23.

PELAYO, I. M. **Geodesic Convexity in Graphs**. New York: Springer-Verlag, 2013.

VIEIRA, M. P. **Uma heurística gulosa para o problema de classificação geodésica.** 2019. Trabalho de Conclusão de Curso (graduação)–Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2019.

ZAKI, M. J.; MEIRA JR, W. **Data mining and analysis**: fundamental concepts and algorithms. New York, USA: Cambridge University Press, 2014. ISBN 0521766338, 9780521766333.