UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DAIANE MENDES DE OLIVEIRA

ARREDONDAMENTO PROBABILÍSTICO PARA O PROBLEMA DE AGM COM RESTRIÇÃO DE GRAU E CENTRAIS E TERMINAIS FIXOS

QUIXADÁ

2017

DAIANE MENDES DE OLIVEIRA

ARREDONDAMENTO PROBABILÍSTICO PARA O PROBLEMA DE AGM COM RESTRIÇÃO DE GRAU E CENTRAIS E TERMINAIS FIXOS

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Orientador: Prof. Dr. Críston Pereira de Souza

QUIXADÁ

2017

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)

O46a Oliveira, Daiane Mendes de.
Arredondamento probabilístico para o problema de AGM com restrição de grau e centrais e terminais fixos / Daiane Mendes de Oliveira. – 2017.
32 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2017.
Orientação: Prof. Dr. Críston Pereira de Souza.

1. Algoritmos. 2. Árvore geradora mínima. 3. Programação inteira. I. Título.

CDD 004

DAIANE MENDES DE OLIVEIRA

ARREDONDAMENTO PROBABILÍSTICO PARA O PROBLEMA DE AGM COM RESTRIÇÃO DE GRAU E CENTRAIS E TERMINAIS FIXOS

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Aprovada em: ___/___/_____

BANCA EXAMINADORA

Prof. Dr. Críston Pereira de Souza (Orientador)
Universidade Federal do Ceará – UFC

Prof. Me. Lucas Ismaily Bezerra Freitas
Universidade Federal do Ceará - UFC

Prof. Dr. Paulo de Tarso Guerra Oliveira
Universidade Federal do Ceará - UFC

AGRADECIMENTOS

Agradeço a Deus por ter sempre me iluminado e me dado forças quando mais precisei, e sei que tudo que conquistei foi por graça Dele.

Agradeço aos meus pais, por terem me dado a educação e apoio necessário para minha formação, aos meus irmãos pela amizade e companheirismo que sempre tivemos.

Agradeço a minha segunda família, que ganhei na graduação, pelo apoio, amizade e companheirismo.

Agradeço aos professores que tive durante esta jornada acadêmica que contribuíram bastante para minha formação, ensinando lições que me fizeram crescer em diversos aspectos.

Agradeço ao professor Críston Souza pela orientação neste trabalho e por ter contribuído na minha formação acadêmica.

Agradeço aos professores participantes da banca examinadora pelo tempo, pelas valiosas colaborações e sugestões.

Agradeço a todos os meus amigos e colegas, por fazerem da minha jornada inesquecível e que trará boas recordações.

A todos que direta ou indiretamente fizeram parte da minha formação.

“Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer.”

(Alan Turing)

RESUMO

Uma árvore é um tipo particular de grafo que possui grande importância no campo de Otimização e Combinatória. Tal importância justifica-se por ser uma estrutura usada para modelar inúmeros problemas ou subproblemas que envolvem conectividade. Um grafo pode ter muitas árvores geradoras, ou seja, uma árvore que contém todos os vértices deste grafo. Se cada aresta do grafo possui um custo, temos que o custo da árvore geradora será a soma dos custos das arestas. Logo, se as arestas tiverem custos diferentes então, no pior caso, cada possível árvore geradora terá custo diferente. O problema consiste em encontrar a árvore geradora de custo mínimo. Esse problema é conhecido como Árvore Geradora Mínima (AGM). Na literatura, existem diversas variações do AGM, onde se impõe restrições adicionais. Em especial, o presente trabalho enfatiza o problema AGM com restrição de grau mínimo. Focamos no Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-MST) (DIAS et al., 2017). Este trabalho tem como objetivo propor e avaliar uma nova solução para o problema MDF-MST, comparando os resultados com modelos em Programação Linear Inteira existentes na literatura. Por meio de testes computacionais foi analisado o custo, em termos de tempo e qualidade da solução. Ao final foram comparadas duas abordagens de solução: i) Programação Linear Inteira (PLI) e ii) Técnica de Arredondamento Probabilístico (AP). Concluímos que a heurística proposta usando a técnica de arredondamento probabilístico apresentou os melhores resultados em termos de tempo para encontrar a solução.

Palavras-chave: Algoritmos. Árvore Geradora Mínima. Programação Linear Inteira. Arredondamento Probabilístico

ABSTRACT

A tree is a particular type of graph that holds great importance in the field of Optimization and Combinatorics. Such importance is justified because it is a framework used to model numerous problems or subproblems involving connectivity. A graph can have many spanning trees, i.e a tree that contains all its vertices of this graph. If each edge of the graph has a cost, we have that the cost of the spanning tree will be the sum of the costs of its edges. So if the edges have different costs, then in the worst case, each possible tree will have a different cost. The problem is to find the minimum-weight spanning tree. This problem is known as Minimum-Weight Spanning Tree (MST). In the literature, there are several variations of the MST, where additional restrictions are imposed. In particular, the present study emphasizes the MST problem with minimum restriction. We focus on the Min-Degree Constrained Minimum Spanning Tree problem with Fixed Centrals and Terminals (MDF-MST) (DIAS et al., 2017). This work aims to propose and evaluate a new solution to the MDF-MST problem, comparing the results with models in Integer Linear Programming existent in the literature. By means of computational tests the cost was analyzed in terms of time and quality of the solution. At the end, two solution approaches were compared: i) Integer Linear Programming (PLI) and ii) Randomized Rounding Approach (AP). We conclude that the proposed heuristic using the randomized rounding approach presented the best results in terms of solution time.

Keywords: Algorithms. Minimum Spanning Tree. Integer Linear Programming. Randomized Rounding

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na modelagem de inúmeros problemas ou subproblemas no campo de Otimização e Combinatória é usada a estrutura de grafos. Um grafo G é um par ordenado (V, E), sendo V um conjunto finito de elementos e E um conjunto de pares não ordenados de elementos pertencentes a V. Chamamos de vértices os elementos que pertencem a V, e de arestas os elementos pertencentes a E.

Uma árvore é um tipo particular de grafo, muito usada para modelar problemas que envolvem conectividade. Uma árvore é um grafo conexo que não contém ciclos. Diz-se que uma árvore A é geradora de um grafo G se A contém todos os vértices de G. Além disso, um grafo com n vértices é uma árvore geradora se somente se é conectado e possui $n-1$ arestas. Como exemplo, se o grafo modela uma rede, então uma árvore geradora compreende um subconjunto minimal de ligações que permitem comunicação entre todos os pares de nós da rede; ou ainda, é uma subestrutura da rede que garante conectividade sem ligações redundantes (MAGNANTI; WOLSEY, 1995).

Um grafo pode ter muitas árvores geradoras. Se cada aresta do grafo possui um custo, temos que o custo da árvore geradora será a soma dos custos das arestas. Logo, se as arestas tiverem custos diferentes então, no pior caso, cada possível árvore geradora terá custo diferente. O problema consiste em encontrar a árvore geradora de custo mínimo. Esse problema é conhecido como Árvore Geradora Mínima (AGM) (*Minimum Spanning Tree*, em inglês, MST).

Na literatura há diversas variantes do AGM, onde se impõem condições adicionais à árvore geradora A, como por exemplo em Salama, Reeves e Viniotis (1997). Uma destas restrições adicionais está relacionada ao grau $d_A(v)$ de cada vértice $v \in V$, isto é, ao número de arestas incidentes a v em A, como pode ser visto em Narula e Ho (1980) e Almeida, Martins e Souza (2012). Entre essas variações o presente trabalho enfatiza o problema AGM com restrição de grau mínimo. Em especial, focamos no Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-MST).

Aplicações práticas para o MDF-MST aparecem em projetos de redes para sistemas com circuito integrado, estradas, redes de energia e computadores, quando se deseja uma rede mínima de conexão com nós especiais onde a informação ou instalações são centralizadas. Esses nós, ditos centrais (que podem ser centros de distribuição ou dispositivos de comunicação centralizados) e os nós terminais (que podem ser clientes ou dispositivos periféricos) devem se conectar, de forma que cada nó central deve se ligar a um número mínimo de outros nós centrais

ou terminais, e estes últimos devem se conectar a um único nó central (DIAS et al., 2013).

O presente trabalho tem como objetivo propor e avaliar novas heurísticas usando a técnica de Arredondamento Probabilístico aplicada ao problema MDF-MST, comparando os resultados com modelos em Programação Linear Inteira existentes na literatura. Por meio de testes computacionais foi analisado o custo, em termos de tempo e qualidade da solução. Foi comparado e avaliado duas abordagens de solução para o problema MDF-MST: i) Programação Linear Inteira (PLI) (CORMEN et al., 2009) (WOLSEY, 1998) e ii) Técnica de Arredondamento Probabilístico (AP) (WILLIAMSON; SHMOYS, 2011).

Este trabalho está organizado da seguinte forma: na Seção 1.1 são apresentados os principais trabalhos relacionados, com uma breve discussão sobre como eles se assemelham e se relacionam com a proposta aqui apresentada; o Capítulo 2 contém um resumo dos principais conceitos utilizados neste trabalho; o Capítulo 3 mostra a descrição das etapas para o desenvolvimento deste trabalho; no Capítulo 4 os resultados são descritos e analisados; e por fim, no Capítulo 5 são apresentados os objetivos alcançados, conclusão e possíveis trabalhos futuros.

1.1 Trabalhos Relacionados

Nesta Seção, apresentamos os trabalhos de Dias et al. (2017) e Gharan, Saberi e Singh (2011) que são os principais contribuidores para o desenvolvimento deste trabalho.

A seguir é apresentado o que cada um destes trabalhos aborda.

1.1.1 A Randomized Rounding Approach to the Traveling Salesman Problem

Gharan, Saberi e Singh (2011) apresentam sobre o problema do caixeiro viajante com métricas no grafo. Tais métricas são definidas pelas menores distâncias do percurso em um grafo qualquer não direcionado. Ou seja, dado um grafo arbitrário não direcionado $G = (V, E)$ deseja-se encontrar o circuito mais curto que visita cada vértice pelo menos uma vez. Com isso, desenvolveram um algoritmo aproximativo, usando a técnica de AP, para o problema.

O algoritmo proposto encontra uma árvore geradora cujo custo é limitado superiormente pelo ótimo. Em casos em que a solução do modelo é quase inteira, o algoritmo seleciona a árvore geradora aleatoriamente por amostragem de uma distribuição de entropia máxima definida pela relaxação da programação linear. Além disso, realizaram análise baseada em uma variedade de ideias e provaram novas propriedades de cortes próximos do mínimo para

grafos arbitrários.

1.1.2 Min-Degree Constrained Minimum Spanning Tree problem with Fixed Centrals and Terminals: complexity, properties and formulations

Dias et al. (2017) provam que a versão de otimização do problema de Árvore Geradora Mínima com restrição de Grau Mínimo com Centrais e Terminais Fixos é NP-Difícil para grafos completos. Apresentam vários modelos em PLI, baseadas em formulações já conhecidas para o problema de Árvore Geradora Mínima, juntamente com uma comparação teórica entre os limites inferiores das relaxações lineares. Além disso, propuseram três heurísticas lagrangianas e realizaram experimentos computacionais comparando em termos de tempo e qualidade da solução os desempenhos das heurísticas e das formulações.

Assim, baseado nos trabalhos citados, o trabalho aqui proposto irá realizar uma avaliação de soluções propostas para o problema MDF-MST, através de testes computacionais analisando o custo em termos de tempo e qualidade da solução. Serão utilizado modelo em PLI, junto com o solver CPLEX, baseado no trabalho Dias et al. (2017). Além disso, este trabalho pretende realizar uma análise com heurísticas aplicando a técnica de AP com base em Gharan, Saberi e Singh (2011). Portanto o foco do presente trabalho é realizar a comparação entre a solução fornecida pelos abordagem em PLI e heurística usando a técnica de AP.

Os trabalhos citados também apresentam diferenças em relação ao presente trabalho. Dias et al. (2017) é o que mais se assemelha a este trabalho, porém não utiliza a técnica AP e Gharan, Saberi e Singh (2011) que apresentam um problema diferente, o problema do caixeiro viajante (em inglês, *Travelling Salesman Problem* - TSP).

Tais diferenças e semelhanças entre os trabalhos citados e o presente trabalho podem ser conferidas na Tabela 1, tal que o cabeçalho da tabela são as características do presente trabalho e para cada trabalho relacionado é preenchido as características com SIM ou NÃO (SIM se o trabalho relacionado possui a característica e NÃO caso contrário).

Tabela 1 – Diferenças e semelhanças dos trabalhos relacionados com o trabalho proposto

	MST	MDF-MST	LPI	AP	Problema
Dias, Campêlo, Souza, e Andrade (2017)	SIM	SIM	SIM	NÃO	MDF-MST
Gharan, Saberi e Singh (2011)	NÃO	NÃO	SIM	SIM	TSP

Fonte – Elaborado pela autora

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, serão abordados os principais conceitos relacionados a este trabalho e qual a contribuição de cada conceito para seu desenvolvimento.

2.1 Árvore Geradora Mínima

Uma árvore é um tipo particular de grafo que possui grande importância no campo de Otimização e Combinatória. Tal importância justifica-se por ser uma estrutura usada para modelar inúmeros problemas ou subproblemas que envolvem conectividade.

Uma árvore é um grafo que não contém ciclos. Uma árvore (ou subárvore) de grafo arbitrário não direcionado $G = (V,E)$ é um subgrafo $A = (V^{'},E^{'})$ acíclico. Diz-se que A é uma árvore geradora de G se A contêm todos os vértices de V, ou seja $V^{'} = V$, de modo que se uma aresta (u,v) for adicionada unindo dois vértices em A cria-se um ciclo único com arestas que já estão na árvore. Além disso, um grafo com n vértices é uma árvore geradora se somente se é conectado e possui $n-1$ arestas (MAGNANTI; WOLSEY, 1995).

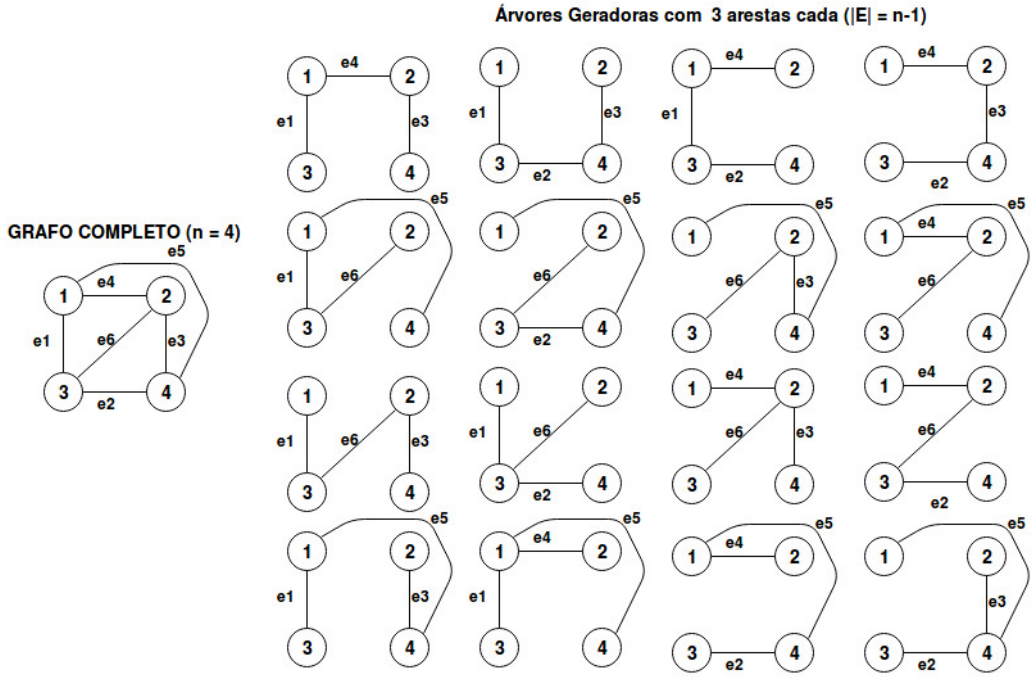
Um grafo pode ter muitas árvores geradoras, em especial para grafos completos existe uma equação para calcular o número de árvores geradoras com n vértices. Esta equação foi descoberta por Cayley (1889) e tornou-se o teorema conhecido como Cayley's Formula. (BONDY; MURTY, 2008)

Definition 2.1.1 *(**Cayley's Formula**). O número de árvores geradoras com n vértices é* n^{n-2}

Por exemplo, para um grafo completo G com quatro vértices ($n = 4$), pela equação temos $4^{4-2} = 16$, logo é possível encontrar dezesseis árvores geradoras, ou seja, dezesseis subgrafos de G com $(n-1)$ arestas cada. Veja a Figura 1 como exemplo.

Se cada aresta do grafo possui um custo, temos que, o custo da árvore geradora será a soma dos custos das arestas. Logo, se as arestas tiverem custos diferentes então, no pior caso, cada possível árvore geradora terá custo diferente. O problema é como encontrar a árvore geradora de custo mínimo. Isto é, dado um grafo $G = (V,E)$ e c_e um custo associado com cada aresta $e \in E$, tal que $c_e \geq 0$, desejamos encontrar uma árvore geradora A em G que minimize o custo total das arestas em A, tal que $\sum_{e \in E} c_e$ seja minimizado. Esse problema é conhecido como Árvore Geradora Mínima (AGM) (*Minimum Spanning Tree*, em inglês, MST).

Figura 1 – Exemplo árvore geradora



Fonte – Elaborado pela autora

Na literatura, existem diversas variações do AGM, onde se impões restrições adicionais. Pode-se citar Salama, Reeves e Viniotis (1997), Narula e Ho (1980), Almeida, Martins e Souza (2012), como exemplos de problemas relacionado ao AGM. Entre essas variações o presente trabalho enfatiza o problema AGM com restrição de grau mínimo. Em especial, focamos no Problema da Árvore Geradora Mínima com Restrição de Grau Mínimo e Centrais e Terminais Fixos (MDF-MST).

O problema MDF-MST generaliza o problema AGM. No MDF-MST, o conjunto de vértices V é particionado em $V = C \cup T$, $C \cap T = \emptyset$, onde os vértices em C são chamados de centrais e têm de satisfazer uma restrição de grau mínimo, e os vértices em T são chamados terminais e devem ter grau 1. Formalmente, é possível descrever o problema MDF-MST da seguinte forma:

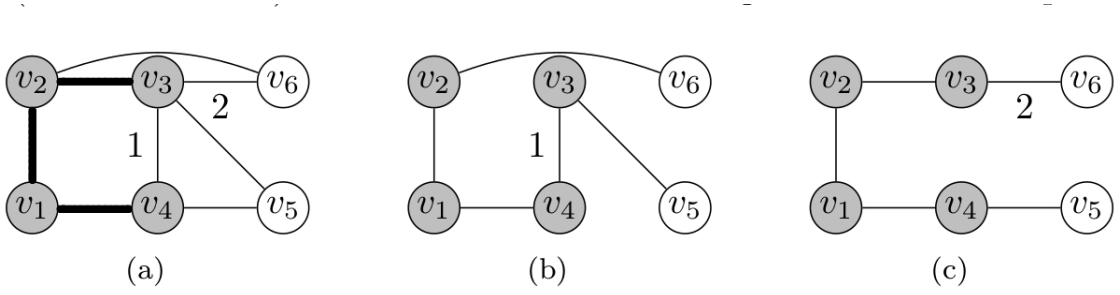
Definition 2.1.2 ***(MDF-MST).*** *Dado $G = (C \cup T, E)$ um grafo simples e conexo, com $c_e \geq 0$ para cada aresta $e \in E$, e seja $d_v \in \mathbb{Z}$ e $d_v \geq 1$, o valor de grau mínimo para cada vértice $v \in C$. Deseja-se encontrar uma árvore geradora mínima A de G, onde $d_A(v) \geq d_v$ para cada $v \in C$ e $d_A(v) = 1$ para cada $v \in T$.*

Uma solução viável para o problema MDF-MST (DIAS et al., 2017) consiste em uma árvore geradora do subgrafo induzido pelos centrais, ou seja o grafo definido pelos os

vértices centrais e todas as arestas que aparecem no grafo original sobre este conjunto de vértices, juntamente com ligações entre terminais e centrais que satisfazem as restrições de grau mínimo. Assim pode-se usar formulações que definam uma árvore geradora e adicionar restrições particulares do problema MDF-MST para obter um modelo para solucioná-lo. Restrições adicionais garantem a conectividade entre centrais e terminais e o grau mínimo para cada central.

Por exemplo, uma instância do MDF-MST pode ser descrita como: $C = \{v_1, v_2, v_3, v_4\}$, $T = \{v_5, v_6\}$ e $d_v = 2$ para todo $v \in C$; o custo de cada aresta é 0 exceto para $c(v_3, v_4) = 1$ e $c(v_3, v_6) = 2$ (ilustrado na Figura (2)), tal que, (a) - Grafo G e AGM de $G(C)$, com linhas mais grossas. (b) - Solução ótima para o MDF-MST. (c) - Solução viável única para o problema AGM de $G(C)$. A remoção da aresta (v_4, v_5) de G não produz nenhuma solução viável usando a AGM de $G(C)$.

Figura 2 – Exemplo de instância MDF-MST



Fonte – Dias et al. (2017)

Como já foi citado, uma solução para o MDF-MST consiste em uma árvore geradora de $G(C)$ e ligações entre terminais e centrais que satisfazem as restrições de grau mínimo. Assim, pode-se obter modelos em PLI para o MDF-MST utilizando formulações que definam uma árvore geradora e adicionando restrições particulares do problema. Restrições adicionais garantem a conectividade entre centrais e terminais e o grau mínimo para cada central.

No presente trabalho foi utilizado modelo em PLI, do trabalho de Dias et al. (2017), para o problema MDF-MST.

2.2 Programação Linear Inteira

Um problema que pode ser formulado com a ação de maximizar ou minimizar um objetivo, sendo dados recursos limitados e restrições concorrentes. E se o objetivo pode ser especificado como uma função linear de certas variáveis e se as restrições lineares podem ser

especificadas sobre recursos como igualdades ou desigualdades sobre essas variáveis, então tem-se um problema de Programação Linear (PL) (CORMEN et al., 2009).

Em um problema de PL, as formulações são feitas com base em variáveis, que representam algum tipo de decisão a ser tomada, chamadas de variáveis de decisão. Elas são delimitadas por inequações e equações lineares denominadas restrições. Uma atribuição de valores reais às variáveis de decisão, de forma que as restrições sejam satisfeitas é chamada de solução viável. Quando existe essa atribuição então diz-se que o problema linear é viável. Já se não existir essa atribuição o programa linear é dito inviável.

Além das restrições, há também a função objetivo, onde o programa em PL procura encontrar uma solução viável que maximize ou minimize esta função objetivo, e esta é chamada de solução ótima. O valor de uma função objetivo para uma certa solução viável qualquer encontrada é chamado de valor daquela solução, e o valor da função objetivo para a solução ótima é chamado de valor do programa em PL. Por fim, diz-se que o programa linear é resolvido se uma solução ótima for encontrada.

Nesse contexto, um modelo em PL é um problema de otimização com uma função objetivo linear, um conjunto de restrições lineares e um conjunto de restrições de intervalo impostas sobre as variáveis de decisão, cujos valores o próprio modelo poderá determinar. Segue um exemplo de um modelo de otimização:

Função Objetivo:

$$\min \quad \sum_{j=0}^{n} (c_j x_j) \tag{2.1}$$

Restrições:

$$\text{sujeito a:} \quad \sum_{j=1}^{n} a_{ij} x_j = b(i) \qquad \forall i \in 1,2,\ldots,m \tag{2.2}$$

$$x_j \geq 0 \qquad \forall j \in 1,2,\ldots,m \tag{2.3}$$

Este modelo em PL possui uma função objetivo de minimização (2.1), n variáveis de decisão não negativas x_j e m restrições de igualdade (2.2). Existem várias maneiras alternativas de se modelar um problema em PL, por exemplo, a função objetivo pode ser colocada na forma de maximização ou as restrições podem ser desigualdades. Um problema está na forma padrão se apresentar restrições de igualdade, variáveis de decisão não negativas e a função objetivo na forma de minimização.

Para um PL existem algoritmos de tempo polinomial em relação ao tamanho do modelo (número de variáveis e restrições), como por exemplo, o ellipsoid e o método de ponto interior. Além desses, existe o simplex que, apesar de não ser polinomial no pior caso, geralmente possui melhores resultados na prática, sendo o mais comumente utilizado (CORMEN et al., 2009).

Um tipo especial de modelo de PL são os que possuem uma restrição adicional de que as suas variáveis possuem valores inteiros. A esse tipo de problema damos o nome de Programação Linear Inteira (PLI), ou simplesmente Programação Inteira. A adição dessa restrição torna o problema NP-Difícil (CORMEN et al., 2009). Podemos ainda ter modelos em que apenas algumas variáveis possuem restrições de integralidade (Programação Inteira Mista), e outros em que as variáveis possuem valores binários 0 ou 1 (Programação Inteira Binária) (WOLSEY, 1998).

Neste trabalho foi comparado um modelo em PLI, para o problema MDF-MST, e uma heurística usando a técnica de Arredondamento Probabilístico, que é aplicável para a classe de algoritmos em PLI.

2.3 Arredondamento Probabilístico

O desenvolvimento de algoritmos de aproximação surgiu em resposta à dificuldade computacional de muitos problemas de otimização combinatória: muitos dos quais são NP-Difíceis. Nessa situação, é razoável sacrificar a otimalidade em troca de uma aproximação de boa qualidade que possa ser calculada em tempo polinomial. Esse compromisso entre perda de otimalidade e ganho em eficiência é o paradigma de algoritmos de aproximação.

Convém observar que um algoritmo de aproximação não é simplesmente uma heurística: ele permite encontrar eficientemente, um elemento do domínio cujo valor guarda uma relação pré-estabelecida com o valor-ótimo (FERREIRA et al., 2001). Temos assim, a seguinte definição para algoritmos de aproximação.

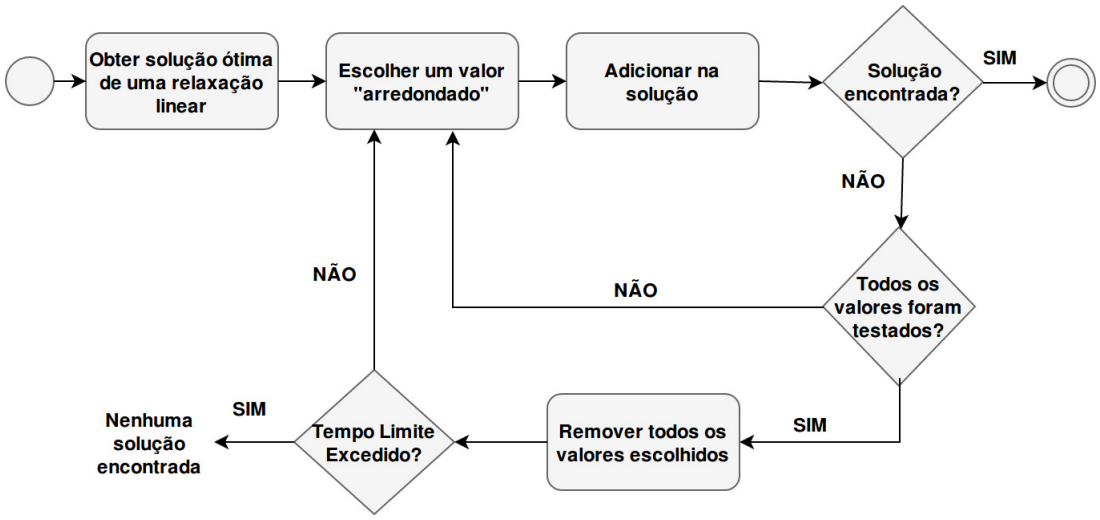
Definition 2.3.1 ***(Algoritmos de aproximação).*** *Um algoritmo de α-aproximação para um problema de otimização é um algoritmo de tempo polinomial que, para todas as ocorrências do problema, produz uma solução cujo valor esteja dentro de um fator α do valor da solução ótima.*

Para um algoritmo de α-aproximação chama-se α a garantia de desempenho do algoritmo (na literatura, também é frequentemente chamado de razão de aproximação ou fator de

aproximação do algoritmo). Adotando a convenção de $\alpha > 1$ para problemas de minimização, enquanto $\alpha < 1$ para problemas de maximização. Assim, um algoritmo de $\frac{1}{2}$-aproximação para um problema de maximização é um algoritmo de tempo polinomial que sempre retorna uma solução cujo valor é pelo menos metade do valor ótimo (WILLIAMSON; SHMOYS, 2011). De forma análoga, um algoritmo de 2-aproximação para um problema de minimização retorna uma solução cujo valor é no máximo o dobro do valor da solução ótima. Por fim, um 1-aproximação é um algoritmo exato para o problema.

Para projetar um algoritmo de aproximação existe uma quantidade significativa de métodos e técnicas. Em particular, o foco deste trabalho é a técnica de Arredondamento Probabilístico (AP), que é aplicável para a classe de algoritmos em PLI. Esta técnica consiste em arredondar uma solução ótima de uma relaxação linear do problema em PL, obtido ao desconsiderarmos as restrições de integralidade das variáveis inteira.

Figura 3 – Técnica de Arredondamento Probabilístico



Fonte – Elaborado pela autora

Na Figura 3 podemos acompanhar o fluxo de como funciona a técnica de AP. Primeiro devemos obter a solução ótima fornecida pela relaxação linear do problema, em seguida escolhe-se um valor arredondado e adiciona-o na solução, a forma de obter esse valor arredondado é específico para cada problema. Logo após, é necessário verificar se uma solução viável foi encontrada, se sim finaliza o algoritmo retornando a solução. Caso contrário, analisa se todos os valores já foram testados, se não, significa que a solução viável ainda pode ser encontrada e segue para escolher o próximo valor arredondado. Se sim, pode significar que a forma de arrendondar não foi adequada e por isso não conseguimos uma solução. Então remove todos os

valores escolhidos e se o tempo limite ainda não tiver excedido segue para uma nova forma de escolher um valor arredondado, se o tempo limite foi excedido finaliza o algoritmo e nenhuma solução foi encontrada.

Nesta seção, como exemplo, a técnica do arredondamento probabilístico será aplicado ao problema de cobertura mínima por conjunto (*MinCC*).

Definition 2.3.2 ***(MinCC).*** *Dados um conjunto finito de elementos $E = \{e_1, e_2, \ldots, e_n\}$, uma coleção finita de subconjuntos desses elementos $S = \{S_1, S_2, \ldots, S_m\}$, tal que $S_j \subseteq E$, e um peso não negativo $w_j > 0$ para cada subconjunto. O objetivo é encontrar uma coleção de subconjuntos com o menor peso possível e que cubra todos os elementos de E.*

Para aplicar a técnica AP, precisa-se de uma formulação em PLI. Assim uma formulação em PLI para o problema *MinCC* é:

$$\min \quad \sum_{j=1}^{m} (w_j x_j) \tag{2.4}$$

$$\text{sujeito a:} \quad \sum_{j:e_i \in S_j = 0}^{n} x_j \geq 1 \qquad \forall i \in 1, 2, \ldots, n \tag{2.5}$$

$$x_j \in \{0, 1\} \qquad \forall j \in 1, 2, \ldots, m \tag{2.6}$$

A variável de decisão x_j representa a escolha do subconjunto, ou seja, $x_j = 1$ se S_j está incluso na solução, e 0 caso contrário. A restrição (2.5) garante que a solução viável produzida pelo algoritmo será realmente um conjunto cobertura e que todo elemento e_i será coberto, pois pelo menos um dos subconjuntos S_j que contém e_1 será selecionado.

O próximo passo é arredondar os valores fracionários da solução ótima, de modo que seja possível obter uma cobertura de E. O arredondamento pode ser feito da seguinte forma: para cada $e \in E$ seja f_e a frequência de e em S, isto é, o número de elementos de S aos quais e pertence. Seja β a maior das frequências. Como S cobre E, temos que, $\beta > 0$. Um algoritmo projetado para este problema é o algoritmo de Hochbaum (HOCHBAUM, 1982 apud FERREIRA et al., 2001) (ilustrado no Algoritmo 1).

O algoritmo 1, escolhe um conjunto Z em S para fazer parte da cobertura se e somente se o valor da variável x_z é pelos menos $\frac{1}{\beta}$. Como cada e em E pertence a no máximo β conjuntos em S, esse fato, juntamente com a restrição (2.5) garante que $x_Z \geq \frac{1}{\beta}$ para algum Z que contem e. Logo C contém algum Z que inclui e, ou seja, C é uma cobertura de E. (FERREIRA et al., 2001) (BORDINI, 2016).

Algoritmo 1: *MinCC* - HOCHBAUM

```
1  inicio
2      C ← ∅
3      x ← Solução ótima obtida pelo modelo PLI (2.4)
4      para cada e ∈ E faça
5          f_e ← A frequência de e em S
6          β ← A maior das frequências
7          Z ← Escolhe um conjunto em S
8          se x_Z ≥ 1/β então
9              C ← C ∪ {Z}
10         fim
11     fim
12     retorne C
13 fim
```

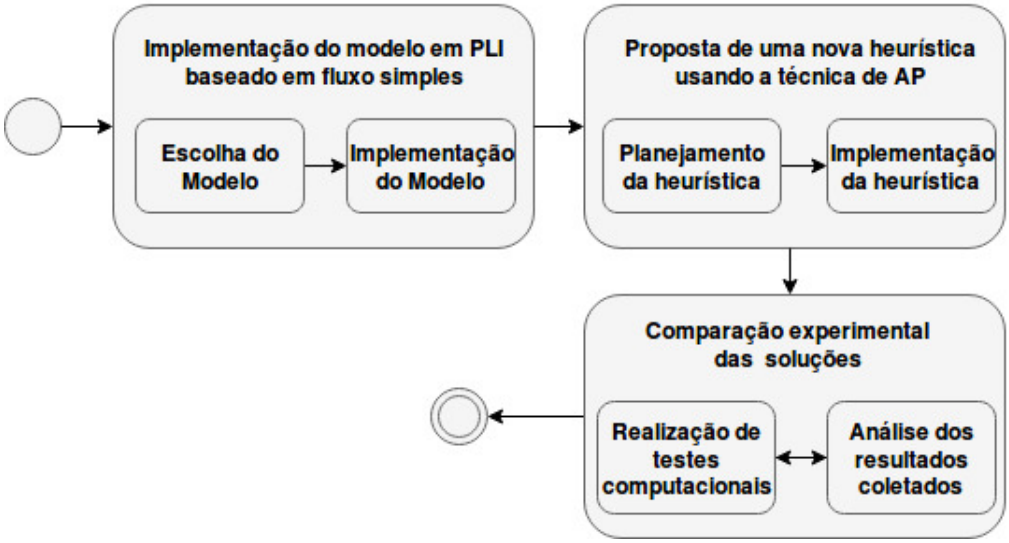
Como as variáveis escolhidas para a solução valem pelo menos $\frac{1}{\beta}$, temos que, no pior caso, ao arrendondá-las para 1 irá aumentá-las no máximo por um fator β. Assim temos que o custo da solução inteira será no máximo β vezes o custo da solução da relaxação linear. Portanto como o custo da relaxação linear é um limite inferior para o custo ótimo, temos que o algoritmo é β-aproximativo.

Este trabalho visa aplicar a técnica AP para o problema MDF-MST, através de heurísticas que dado a solução ótima do modelo em PLI retorne uma árvore geradora A.

3 DESENVOLVIMENTO

O trabalho foi planejado e desenvolvido em três etapas (Figura 4): i) implementação do modelo em PLI baseado em fluxo simples de Dias et al. (2017) ii) proposta de uma nova heurística usando a técnica de AP para o problema MDF-MST; iii) comparação experimental das soluções, analisando o custo, em termos de tempo e qualidade da solução. Na Figura 4 pode-se acompanhar em detalhes os passos seguidos para o desenvolvimento deste trabalho.

Figura 4 – Fluxograma das etapas de desenvolvimento do trabalho.



Fonte – Elaborado pela autora

3.1 Implementação do modelo em PLI baseado em fluxo simples

Inicialmente foi implementado o modelo em PLI, baseado em fluxo simples direcionado, MDF-MST$^{r}_{flo}$ apresentado em Dias et al. (2017). Este modelo foi escolhido por usar um número polinomial de variáveis e restrições em relação à entrada, além de ser um modelo simples e de fácil entendimento. Antes de apresentar a formulação do deste modelo torna-se necessário definir algumas notações e variáveis usadas no modelo.

Seja G um grafo simples, $V(G)$ e $E(G)$ é chamado de conjunto de vértices e conjunto de arestas, respectivamente. Seja $U^{'}$, $V^{'} \subseteq V(G)$, logo $\delta(U^{'},V^{'})$ são as arestas com uma extremidades em $U^{'}$ e outra em $V^{'}$, ou seja, $\delta(U^{'},V^{'}) = \{uv \in E(G) : u \in U^{'} \text{ e } v \in V^{'}\}$. Por simplicidade, usa-se $\delta(U^{'}) = \delta(U^{'},V \setminus U^{'})$ e $\delta(v) = \delta(\{v\})$. Se atribuir orientação as arestas de G usa-se a notação $\delta^{+}(v)$ para nos referimos aos vértices que saem de v, e $\delta^{-}(v)$ para nos

referimos aos vértices que chegam em v. O grau de um vértice v em G é $d(v) = |\delta(v)|$. E por último tem-se a variável r que representa a raiz da árvore.

Na definição do problema MDF-MST, considera-se $G = (C \cup T, E)$, como um grafo simples e conexo, tal que C e T são o conjunto de centrais e terminais, respectivamente. Por último temos que, o subgrafo A de G induzido pelos centrais é denotado por $A(C) = (C, E(C))$, onde $E(C) = \delta(C,C)$. A seguir pode-se acompanhar a formulação do modelo MDF-MST$^r_{flo}$. MDF-MST$^r_{flo}$:

$$\text{min} \quad \sum_{e \in E(G)} (c_e x_e) \tag{3.1}$$

$$\text{sujeito a:} \quad \sum_{e \in \delta(i)} x_e \geq d_i \qquad \forall i \in C \tag{3.2}$$

$$\sum_{e \in \delta(i)} x_e = 1 \qquad \forall i \in T \tag{3.3}$$

$$\sum_{(i,j) \in \delta^-(j) \cap A(C)} x_{ij} = 1 \qquad \forall j \in C \backslash r \tag{3.4}$$

$$\sum_{(j,i) \in \delta^-(i)} f_{ji} - \sum_{(i,j) \in \delta^+(i)} f_{ij} = 1 \qquad \forall i \in C \backslash r \tag{3.5}$$

$$x_{ij} \leq f_{ij} \leq (|\, C \,| - 1) x_{ij} \qquad \forall (i,j) \in A(C) \tag{3.6}$$

$$x_{ij} + x_{ji} \in \{0,1\} \qquad \forall (i,j) \in E \tag{3.7}$$

$$x_{ir} = 0 \qquad \forall (i,r) \in A(C) \tag{3.8}$$

$$f_{ij} \geq 0 \qquad \forall (i,j) \in A(C) \tag{3.9}$$

As restrições (3.2) e (3.3) garantem os graus dos vértices centrais e terminais, as restrições (3.4) - (3.6) são restrições da formulação original para o problema de árvore geradora mínima, ou seja, garantem que seja gerada uma árvore, e as restrições (3.7) - (3.9) indicam o domínio das variáveis.

Para replicação do modelo MDF-MST$^r_{flo}$, apresentado em Dias et al. (2017) foi utilizado a linguagem C++, por conta do seu alto desempenho, juntamente com a biblioteca do o solver CPLEX da IBM.

3.2 Proposta de uma nova heurística usando a técnica de AP

Nesta etapa foi proposto uma nova solução para o problema MDF-MST, usando a técnica de Arredondamento Probabilístico (AP), aqui chamada de heurística AP$^r_{flo}$. A proposta de solução segue o seguinte processo: primeiro recebe como entrada a solução ótima da relaxação

linear, obtida pela relaxação linear do modelo MDF-MST$^r_{flo}$. Em seguida, para o processo de arredondar, realiza-se a normalização desta solução.

O processo de normalização é determinado de acordo com os seguintes passos. Dado o conjunto de arestas x_e com seus respectivos valores, entre 0 e 1, obtidos na relaxação linear do modelo MDF-MST$^r_{flo}$, primeiro realiza-se a soma acumulada deste valores (*sum*). De modo que, dado o valor da aresta na posição i ($x_e[i]$), no vetor de soma acumulada na posição i, ou seja $sum[i]$ possui o somatório dos valores das arestas da primeira posição até a aresta da posição i. Em seguida para cada aresta $e \in x_e$, seu novo valor é determinado pela Equação 3.10.

$$x_{eN}[i] = \frac{x_e[i]}{sum} \tag{3.10}$$

Logo após inicia-se um processo iterativo até que seja obtida uma árvore geradora, ou seja uma solução viável. O procedimento começa realizando o sorteio da aresta e, de modo que a probabilidade de e seja proporcional ao seu valor normalizado. Em seguida, caso e não forme ciclo, adiciona-se e como parte da solução.

Veja o pseudocódigo genérico para a heurística AP$^r_{flo}$ no Algoritmo 1.

Algoritmo 2: HEURÍSTICA AP$^r_{flo}$

```
1  inicio
2      A ← ∅
3      x_e ← Solução ótima obtida pelo modelo PLI
4      x_eN ← Normalização de x_e
5      para cada i ∈ {1,2,...,|E|} faça
6          e ← Sorteia aresta com probabilidade proporcional ao x_eN[i]
7          se e não forma ciclo em A então
8              A ← A ∪ {e}
9          fim
10     fim
11     retorne A
12 fim
```

Fonte – Elaborado pela autora.

Esta heurística se assemelha ao algoritmo Kruskal (para o problema AGM), de modo que o conjunto A é uma floresta. A aresta escolhida para ser adicionada a A é sempre uma aresta no grafo que conecta dois componentes distintos.

3.3 Comparação experimental das soluções

Para a etapa de comparação, foram usadas as instâncias ALM, originalmente geradas por Martins e Souza (2009), com diferentes tipos de entradas, apenas variando os parâmetros, como o número de centrais e o tamanho do grafo. Cada uma das instâncias ALM é um grafo tal que os vértices correspondem a pontos gerados aleatoriamente no plano euclidiano dentro de um retângulo de dimensões $480x640$. O custo de cada aresta (u,v) é dado pela distância euclidiana entre os pontos u e v. Em cada caso, determina-se o números de vértices centrais c, de terminais $n-c$, onde n é o número total de vértices, e cada vértice central possui um valor de grau mínimo $d(i)$.

As instância foram divididas em quatro grupos, de acordo com o número de centrais, sendo que os grupos compreende aos grafos com 60, 100, 200 e 300 vértices centrais. Na Tabela 2 está listado os conjuntos de instâncias usado nos testes, num total de 16 instâncias testadas.

Tabela 2 – Instâncias da classe ALM para testes

Instâncias ALM					
Nome	n	c	Nome	n	c
ALM60-1	146	60	ALM200-1	482	200
ALM60-2	140	60	ALM200-2	462	200
ALM60-3	122	60	ALM200-3	402	200
ALM60-4	116	60	ALM200-4	382	200
ALM100-1	242	100	ALM300-1	722	300
ALM100-2	232	100	ALM300-2	692	300
ALM100-3	202	100	ALM300-3	602	300
ALM100-4	192	100	ALM300-4	572	300

Fonte – Elaborado pela autora

Nessa fase, também houve a preparação do servidor em que os testes foram executados e definição do tempo limite de duas horas e meia para execução de cada instância de teste. Após a preparação do servidor, o modelo MDF-MST$^r_{flo}$ e a heurística AP$^r_{flo}$ foram submetidos a testes com o intuito de compará-los em relação ao tempo e a qualidade da solução. A qualidade da solução é dada por GAP $= (VS - OPT)/OPT$, onde VS é o valor da solução obtido ao rodar uma instância e OPT é o valor da solução ótima para esta instância. E por fim, os resultados dos testes foram analisados com o propósito de descobrir o comportamento da heurística proposta para o problema.

4 RESULTADOS

Utilizando a linguagem C++ e a biblioteca do solver CPLEX da IBM, foi implementado o modelo MDF-MST$^r_{flo}$ de Dias et al. (2017). Para implementar a heurística AP$^r_{flo}$ foi usada a linguagem em C++ e o módulo Rccp, um pacote que fornece funções em R e oferece uma integração de R e C++. Foi usada a função "sample"do Rcpp para realizar o sorteio das arestas de acordo com os valores da solução da relaxação linear do modelo MDF-MST$^r_{flo}$.

Os testes foram realizados em um servidor com processador de 3,0GHz octacore, 8GB de RAM e executando o sistema operacional Ubuntu 16.04. Foram utilizadas as instâncias ALM de Martins e Souza (2009), determinadas por grafos completos com custos positivos nas arestas e todas definidas no plano euclidiano. Foi estipulado o tempo de duas horas e meia, tempo limite para cada instância, para a execução dos testes, sendo abortadas as que não conseguissem executar nesse tempo.

A seguir os resultados coletados podem ser visualizados na Tabela 3 e logo após uma análise detalhada dos resultados exibidos na tabela.

Tabela 3 – Resultados do Modelo MDF-MST$^r_{flo}$ e da Heurística AP$^r_{flo}$

	MDF-MST$^r_{flo}$		AP$^r_{flo}$	
Instâncias	Tempo(s)	(GAP %)	Tempo(s)	(GAP %)
ALM60-1	1285.49	0.00%	15.119	0.00%
ALM60-2	828.47	0.00%	32.651	0.00%
ALM60-3	488.14	0.00%	15.358	0.00%
ALM60-4	401.80	0.00%	48.284	0.00%
ALM100-1	7644.6	0.00%	803.49	0.00%
ALM100-2	1732.58	0.00%	326.84	0.00%
ALM100-3	5824.57	0.00%	203.07	0.00%
ALM100-4	3947.58	0.00%	373.32	0.00%
ALM200-1	timeout	1.60%	timeout	0.01%
ALM200-2	timeout	1.86%	2103.4	0.00%
ALM200-3	timeout	0.87%	159.12	0.00%
ALM200-4	timeout	7.09%	403.23	0.00%
ALM300-1	30.01	Falta de memória	5612.9	0.00%
ALM300-2	694.79	Falta de memória	6123.9	0.00%
ALM300-3	timeout	2.89%	2325.6	0.00%
ALM300-4	timeout	7.48%	1866.0	0.00%

Fonte – Elaborado pela autora

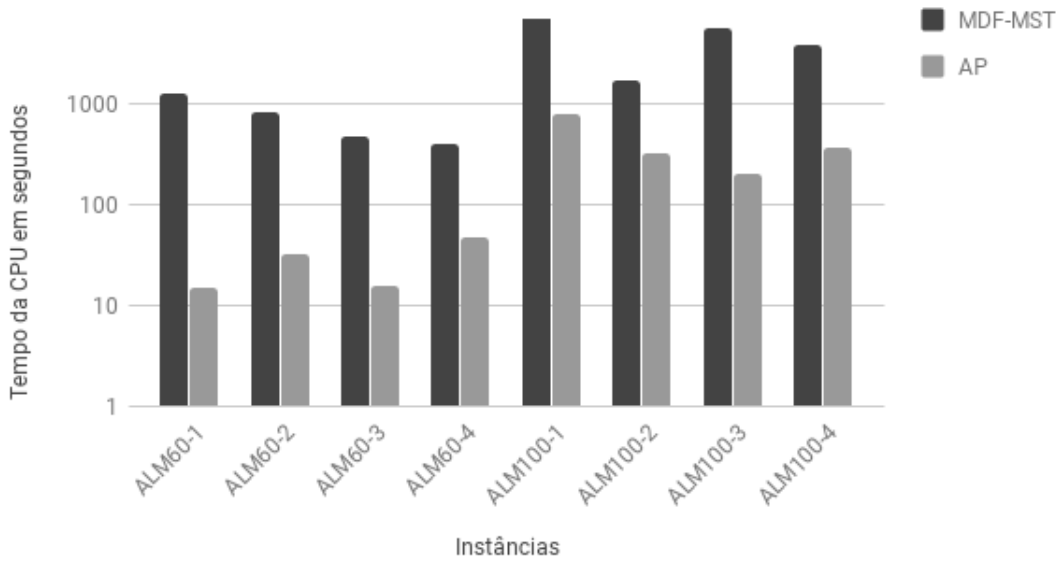
Para facilitar a análise, os resultados foram divididos em três casos e analisados separadamente. Segue a descrição para cada caso analisado: i - instâncias que apresentaram solução ótima nos resultados de ambos os algoritmos, do modelo MDF-MST$^r_{flo}$ e da heurística AP$^r_{flo}$; ii - instâncias que estourou o tempo limite e portanto não conseguiu encontrar a solução

ótima, em um dos algoritmos ou em ambos, e por fim, iii - instâncias que apresentaram falta de memória em pouco tempo de execução no modelo MDF-MST$^r_{flo}$ e não conseguiu encontrar nenhuma solução.

Ao analisar os resultados do caso i, tem-se que metade das instâncias testadas encontram-se neste caso, ou seja foi encontrada a solução ótima em ambos algoritmos testados. Este conjunto de instância corresponde a primeira metade de instâncias que são os grafos cujo a quantidade de vértices são as menores. Para esta primeira metade não faz sentido comparar a qualidade da solução, visto que, o resultado para todas estas instâncias foi a solução ótima. Mas ao observar seus resultados em razão de tempo (segundos) que gastou para encontrar a solução, pode-se concluir que a heurística AP$^r_{flo}$ apresenta os melhores resultados.

Para facilitar a visualização na comparação em termos de tempo entre o modelo MDF-MST$^r_{flo}$ e da heurística AP$^r_{flo}$, na primeira metade das foi criado o gráfico da Figura 5. O eixo y do gráfico está em escala logarítmica para melhor visualização da variação de valores.

Figura 5 – Gráfico de comparação em termos de tempo entre o modelo MDF-MST$^r_{flo}$ e a heurística AP$^r_{flo}$.



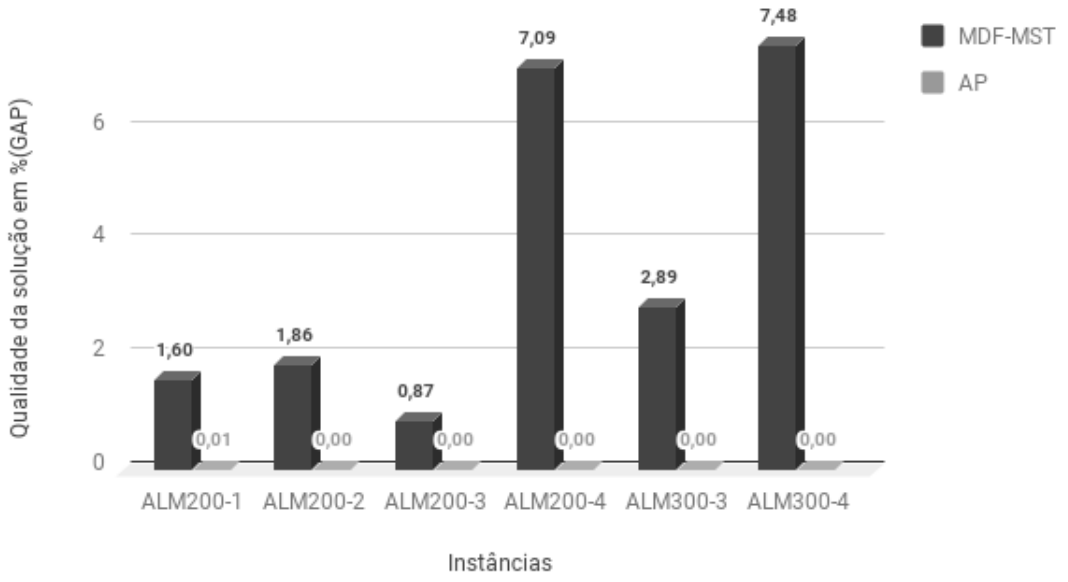
Fonte – Elaborado pela autora

Para os resultados do caso ii, instâncias que estouram o tempo limite e portanto não conseguiram encontrar a solução ótima, em um dos algoritmos ou em ambos. Este conjunto tem um total de seis instâncias e para seus resultados não faz sentido analisar em termos de

tempo, visto que para todas as instâncias deste conjunto o modelo MDF-MST$^r_{flo}$ estourou o tempo. Porém tem-se o segundo dado a analisar, a qualidade das soluções encontradas até o tempo limite. Vale ressaltar que para este conjunto de instâncias ao observar os resultados da heurística AP$^r_{flo}$ apenas uma instância apresentou estouro de tempo e para todas as outras, deste conjunto, foi encontrado a solução ótima.

A seguir para melhor visualização na comparação em termos de qualidade da solução entre o modelo MDF-MST$^r_{flo}$ e a heurística AP$^r_{flo}$, para o conjunto de instâncias do caso ii, foi criado o gráfico da Figura 6.

Figura 6 – Gráfico de comparação em termos da qualidade da solução (GAP) do modelo MDF-MST$^r_{flo}$ e da heurística AP$^r_{flo}$.



Fonte – Elaborado pela autora

E por fim tem-se as instâncias do caso iii, instâncias que apresentaram falta de memória em pouco tempo de execução e não conseguiu encontrar nenhuma solução viável. Neste caso encontram-se duas instâncias, a ALM300-1 e a ALM300-2, em especial são as instâncias cujo os grafos são os maiores em quantidade de vértices. Uma possível causa para a falta de memória nos resultados do modelo MDF-MST$^r_{flo}$ foi a forma como ele foi implementado, é possível que esteja sendo adicionado variáveis demais ao modelo, variáveis desnecessárias. Porém, para estas duas instâncias, a relaxação linear conseguiu retornar uma solução que foi usada como entrada da heurística AP$^r_{flo}$ e que conseguiu encontrar a solução ótima dentro do

tempo estimado.

É importante ressaltar que para realizar uma comparação mais justa, o modelo MDF-MST$^{r}_{flo}$ teria que receber como parâmetro o valor da solução ótima para o PLI, visto que a heurística AP$^{r}_{flo}$ recebe este valor ótimo e o usa como condição de parada. Utilizando esta informação, o modelo MDF-MST$^{r}_{flo}$ poderia também interromper a busca em menos tempo ao comparar a solução atual com o limite inferior recebido como entrada (valor da solução ótima). É possível que este seja o motivo para a diferença na comparação de tempo. Porém esta conclusão só foi constatada no final do trabalho. Portanto, fica como trabalho futuro realizar esta correção e executar novamente os experimentos.

De modo geral ao comparar os resultados, conclui-se que, para as instâncias utilizadas nos testes, a heurística AP$^{r}_{flo}$ tem melhores resultados do que o modelo MDF-MST$^{r}_{flo}$. Pois os resultados, para todas as instâncias, com exceção da ALM200-1, que excedeu o tempo limite, foi encontrado a solução ótima e com os melhores tempos dos testes rodados com a heurística AP$^{r}_{flo}$. Em comparação com o modelo MDF-MST$^{r}_{flo}$ que excedeu o tempo limite para pouco menos da metade das instâncias e apresentou falta de memória para duas instâncias. Já em termos de qualidade da solução entre o modelo MDF-MST$^{r}_{flo}$ e da heurística AP$^{r}_{flo}$, sempre que não foi atingido o tempo limite ou tenha apresentado falta de memória, em ambos foi encontrada a solução ótima.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve o objetivo de avaliar uma nova heurística para o problema MDF-MST (AP^r_{flo}), uma heurística usando a técnica de AP, assim como comparar a nova solução com solução já existente na literatura, o modelo em PLI MDF-MST^r_{flo} de (DIAS et al., 2017) para o problema. Conseguimos alcançar o objetivo do trabalho, sendo que foi proposto e implementado uma nova heurística usando a técnica AP, um modelo em PLI foi replicado para comparação dos resultados e a heurística apresentou os melhores resultados.

Porém para comparar as duas abordagem de solução foram usadas 16 instâncias (Tabela 2), sendo a maior delas um grafo com 722 vértices, tais instâncias são consideradas pequenas no contexto de problemas reais. Para realizar testes com instâncias maiores, similares a de problemas reais, viu-se a necessidade de melhorar ou usar outro modelo para relaxação linear, resultado usado como entrada da heurística. Como não se conseguiu realizar testes mais completos utilizando o modelo MDF-MST^r_{flo} (DIAS et al., 2017), pois houve falta de memória ao testar as instâncias maiores. Um motivo foi a modelagem usada, que declarou muitas variáveis desnecessárias. Um passo futuro para a continuação deste trabalho deve ser melhorar a modelagem ou utilizar outro modelo.

Em Dias et al. (2017) foram realizadas implementações de cinco modelo em PLI, dentre eles o modelo de corte direcionado apresentou bons resultados e se for replicado de forma otimizada consegue processar grandes grafos, mais parecidos com de instâncias de problemas reais. Além disso, a heurística aqui proposta, internamente, faz uso de algoritmo para o problema de AGM. Neste a heurística foi implementada usando a ideia do algoritmo Kruskal, mas uma ideia interessante também seria implementar a heurística usando a ideia do algoritmo Prim. Uma direção de trabalhos futuros é escolher o modelo PLI que apresenta melhores resultados para o problema no contexto de problemas reais, implementar uma segunda heurística usando a ideia do algoritmo Prim e realizar os testes de comparação.

REFERÊNCIAS

ALMEIDA, A. M. de; MARTINS, P.; SOUZA, M. C. de. Min-degree constrained minimum spanning tree problem: complexity, properties, and formulations. **International Transactions in Operational Research**, Wiley Online Library, v. 19, n. 3, p. 323–352, 2012.

BONDY, J.; MURTY, U. **Graph theory (graduate texts in mathematics)**. [S.l.]: Springer New York, 2008.

BORDINI, C. F. **Técnicas probabilísticas aplicadas em algoritmos de aproximação**. 2016.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to algorithms third edition**. [S.l.]: The MIT Press, 2009.

DIAS, F. C.; CAMPêLO, M.; SOUZA, C.; ANDRADE, R. Min-degree constrained minimum spanning tree problem with fixed centrals and terminals: Complexity, properties and formulations. **Computers & Operations Research**, Elsevier, v. 84, p. 46–61, 2017.

DIAS, F. C. S.; ANDRADE, R. C. de; CAMPÊLO, M.; SOUSA, C. P. de. Problema de Árvore geradora mínima com restrição de grau mínimo e centrais fixos1. **XLV Simpósio Brasileiro de Pesquisa Operacional - SBPO, Anais...**, Natal - RN, 2013.

FERREIRA, C.; FERNANDES, C.; MIYAZAWA, F.; SOARES, J.; JR, J. P.; GUIMARÃES, K.; CARVALHO, M.; CERIOLI, M.; FEOFILOFF, P.; DAHAB, R. et al. Uma introdução sucinta a algoritmos de aproximação. **Colóquio Brasileiro de Matemática-IMPA, Rio de Janeiro-RJ**, 2001.

GHARAN, S. O.; SABERI, A.; SINGH, M. A randomized rounding approach to the traveling salesman problem. In: IEEE. **Foundations of Computer Science (FOCS), 2011 IEEE 52nd, Annual...** [S.l.], 2011. p. 550–559.

HOCHBAUM, D. S. Approximation algorithms for the set covering and vertex cover problems. **SIAM Journal on computing**, SIAM, v. 11, n. 3, p. 555–556, 1982.

MAGNANTI, T. L.; WOLSEY, L. A. Optimal trees. **Handbooks in operations research and management science**, Elsevier, v. 7, p. 503–615, 1995.

MARTINS, P.; SOUZA, M. C. de. Vns and second order heuristics for the min-degree constrained minimum spanning tree problem. **Computers & Operations Research**, Elsevier, v. 36, n. 11, p. 2969–2982, 2009.

NARULA, S. C.; HO, C. A. Degree-constrained minimum spanning tree. **Computers & Operations Research**, Elsevier, v. 7, n. 4, p. 239–249, 1980.

SALAMA, H. F.; REEVES, D. S.; VINIOTIS, Y. The delay-constrained minimum spanning tree problem. In: IEEE. **Computers and Communications, 1997. Proceedings., Second IEEE Symposium on**. [S.l.], 1997. p. 699–703.

WILLIAMSON, D. P.; SHMOYS, D. B. **The design of approximation algorithms**. [S.l.]: Cambridge university press, 2011.

WOLSEY, L. A. **Integer programming**. [S.l.]: Wiley New York, 1998. v. 42.