UNIVERSIDADE FEDERAL DO CEARÁ CAMPUS DE RUSSAS

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

KAINAN FARIA DE OLIVEIRA

UMA ABORDAGEM HEURÍSTICA AO PROBLEMA DE INFECÇÃO EM GRAFOS SUBMETIDO À REGRAS P3 E P4

RUSSAS

2019

KAINAN FARIA DE OLIVEIRA

UMA ABORDAGEM HEURÍSTICA AO PROBLEMA DE INFECÇÃO EM GRAFOS SUBMETIDO À REGRAS P3 E P4

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal do Ceará Campus de Russas, como requisito parcial à obtenção do título de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marcio Costa Santos

RUSSAS

2019

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)

O47a Oliveira, Kainan Faria de.
Uma abordagem heurística ao problema de infecção em grafos submetido à regras P3 e P4 / Kainan Faria de Oliveira. – 2019.
43 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2019.
Orientação: Prof. Dr. Marcio Costa Santos.

1. Otimização. 2. Infecção de Grafos. 3. Regras de infecção. I. Título.

CDD 005

KAINAN FARIA DE OLIVEIRA

UMA ABORDAGEM HEURÍSTICA AO PROBLEMA DE INFECÇÃO EM GRAFOS SUBMETIDO À REGRAS P3 E P4

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal do Ceará Campus de Russas, como requisito parcial à obtenção do título de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Marcio Costa Santos (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bonfim Amaro Júnior
Universidade Federal do Ceaŕa (UFC)

Prof. Ms. Tatiane Fernandes Figueiredo
Universidade Federal do Ceaŕa (UFC)

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE FIGURAS

AGRADECIMENTOS

Ao Prof. Dr. Márcio Santos Costa por me orientar em meu trabalho de conclusão de curso.

A todos os amigos que construíram para quem de alguma forma esse momento fosse possível.

Ao grupo de pesquisa Nemo, por proporcionar uma maior obtenção de conhecimentos me preparando para ser um melhor profissional.

Aos meus pais, irmão, avó e tia, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Agradeço a mim, por mesmo com todas as dificuldades e todas as vontades de desistir, mantive o objetivo principal em mente, fazendo com que nunca desistisse.

“O que realmente importa na vida não é o destino
e sim a jornada.”

(Ted Mosby)

RESUMO

O problema de infecção em grafos, pode ser visto como um conjunto de vértices infectados S espalhando um vírus em um grafo $G = (V, E)$. Nos problemas de infecção em grafos, vértices não infectados serão infectados em um tempo i se respeitarem as regras de infecção previamente definidas de acordo com o problema no tempo $i - 1$ (DRAIEF; GANESH, 2010). Foi estudado o problema de determinar o menor conjunto de vértices infectados do grafo que consiga infectar todo o grafo. É proposto um modelo matemático relativo a cada regra de infecção estudada, no caso, P3 e P4, depois partiu-se para a implementação dos algoritmos utilizando a biblioteca PuLP da linguagem de programação *Python* para manuseio do solver ILOG IBM CPLEX. Ao final das implementações, pode-se perceber que o modelo obteve resultados promissores. Também é apresentado uma heurística construtiva para o problema que também foi testada com resultados promissores, obtendo muitas vezes, resultados que superam os valores obtidos nos algoritmos exatos implementados, em relação a instâncias com quantidades elevadas de vértices.

Palavras-chave: Grafos. Otimização. Modelo matemático.

ABSTRACT

The graph infection problem can be seen as a set of S infected vertices spreading a virus on a graph $G = (V, E)$. In graph infection problems, uninfected vertices will be infected at a time i if they comply with the infection rules previously defined according to the time problem $i-1$ (DRAIEF; GANESH, 2010). We studied the problem of determining the smallest set of infected graph vertices that can infect the entire graph. A mathematical model is proposed for each infection rule studied, in this case, P3 and P4, and then we started to implement the algorithms using the *Python* programming language PuLP library for handling the ILOG IBM CPLEX solver. At the end of the implementations, it can be seen that the model obtained promising results. Also presented is a constructive heuristic for the problem that was also tested with promising results, often obtaining results that exceed the values obtained in the optimal algorithms implemented, in relation to instances with high quantities of vertices..

Keywords: Graphs. Optimization. Mathematical model

SUMÁRIO

1 INTRODUÇÃO

Cotidianamente usa-se grafos para modelar diversos problemas, muitas vezes existe interesse nos caminhos que ligam seus vértices, seja como um caminho a ser percorrido, um enlace em uma rede de computadores, uma comunicação em banco de dados, entre outros. Neste contexto, um problema que se apresenta é relativo a propagação de falhas ou erros, uma maneira de modelar essas falhas é através de infecção em grafos. Neste trabalho serão analisado dois modelos de infecção de grafos utilizando regras de infecção P3 e P4, assim como uma heurística construtiva para o problema.

O problema de infecção em teoria dos grafos, pode ser visto como um conjunto de vértices infectados S espalha um vírus em um grafo $G = (V, E)$. Nesses problemas de infecção em grafos, tem-se que um vértice não infectado v, em um tempo i, só será infectado se respeitar as regras de infecção previamente definidas de acordo com o problema no tempo i-1.

Neste trabalho é tratado o problema de infecção relativo a determinar o menor conjunto de vértices que infecta todos os vértices do grafo, independente do tempo que a infecção leve para se espalhar, de acordo com uma regra de infecção. O problema de infecção é um tema muito abordado em teoria dos grafos, mas sob o prisma de otimização este tema é pouco estudado, apesar de sua grande possibilidade de utilização em uma vasta gama de aplicações.

Por exemplo, o problema pode ser aplicado em rede de computadores, onde segundo Hosseinalipour *et al.* (2018), a partir de um grafo pode ser feita uma análise, podendo distinguir se a condição que o grafo apresenta é uma infecção que será transmitida a todos os nós pertencentes a rede de computadores ou apenas um conjunto de falhas aleatórias nessa rede, essa análise utiliza abordagens eficazes, podendo apresentar dificuldades pela irregularidade da rede analisada.

Além disso, infecção de grafos pode ser aplicado no planejamento e tratamento da disseminação de epidemias, como mostra Seibold e Callender (2016) onde é modelado um sistema de transmissão de doenças a partir de chamadas redes de contatos, permitindo uma exploração dos dados biológicos e efeitos estocásticos comparando com os modelos clássicos de equações diferenciais ordinárias.

Este trabalho, apresenta duas formulações desenvolvidas para o problema de infecção de grafos utilizando regra de infecção $P3$ e $P4$, assim como, testes realizados com ambos. Além disso, são propostas duas heurísticas construtivas para o problema de infecção, também para as regras de infecção citadas acima trazendo também os testes realizados com a heurística referente a regra $P3$ onde há um comparativo, verificando as diferenças entre os dois métodos. A heurística

*P*4 não foi testada por questões de tempo.

Na Seção 2 é apresentada as preliminares para entendimento do trabalho; A Seção 3 mostra a problemática deste trabalho; A Seção 4 apresenta os trabalhos relacionados mais relevantes para esta pesquisa. Na Seção 5 é discorrido sobre a amostra das contribuições alcançadas com o trabalho; Na Seção 6 é explicado os resultados obtidos; A Seção 7 apresenta as conclusões finais sobre este trabalho.

2 PRELIMINARES

A seguir serão apresentados alguns conceitos básicos de teoria dos grafos e pesquisa operacional para o entendimento do trabalho, além de uma breve descrição sobre heurísticas.

2.1 Conceitos de teoria dos grafos

Em teoria do grafos, um grafo $G = (V, E)$ é definido como um conjunto de elementos denominados vértices, denotado por $V(G)$, e um conjunto de pares não ordenados desses vértices chamado de conjunto de arestas e denotado por $E(G)$. Dados dois vértices v e u, se o par $uv \in E$, dizemos que uv é uma aresta e que v é adjacente ou vizinho de u e vice-e-versa. Alguns conceitos de teoria dos grafos são de fundamental importância para o entendimento do problema de infecção de grafos.

Dado um grafo $G = (V, E)$, uma sequência de vértices $v_1, v_2, ..., v_n$, é dita um caminho se para todo $1 \leq i \leq n-1$ tem-se que $v_i v_{i+1} \in E(G)$. O tamanho de um caminho pode ser obtido pela quantidade total de arestas neste caminho (SANTOS, 2017).

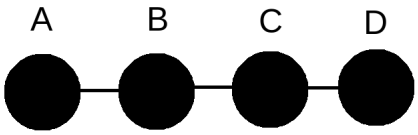

Figura 1 – Caminho de A até D
Fonte: Autor

Um ciclo pode ser definido como sendo um caminho onde o vértice final é o mesmo inicial. Da mesma forma de caminhos, o comprimento de um ciclo se da pela soma de todas as arestas que compõe o ciclo (SANTOS, 2017).

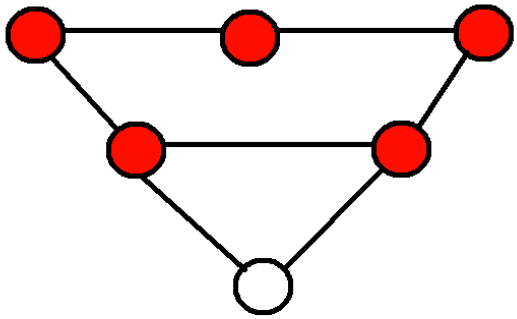

Figura 2 – Grafo com clique de tamanho 3 em vermelho
Fonte: Autor

Dado um grafo $G = (V, E)$, é dito que $A \subseteq V$ é uma clique se seus vértices são dois a dois adjacentes, ou seja, todos os pares de vértices possuem uma aresta que os liga (SOUSA, 2015). A cardinalidade da clique é indicada pelo número de vértices e $\omega(G)$ denota a cardinalidade de uma clique máxima de G.

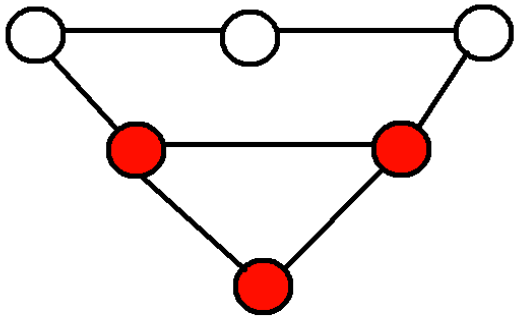

Figura 3 – Grafo com clique de tamanho 3 em vermelho
Fonte: Autor

Um conjunto estável pode ser definido como sendo um subconjunto de vértices dois a dois não adjacentes (SOUSA, 2015). A cardinalidade do conjunto estável é indicada pelo número de vértices no conjunto e $\alpha(G)$, para um grafo $G = (V, E)$, denota a cardinalidade de um conjunto estável máximo.

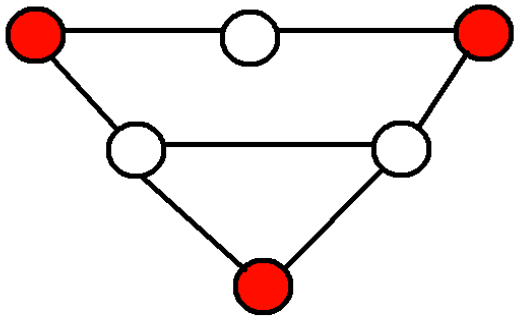

Figura 4 – Grafo com clique de tamanho 3 em vermelho
Fonte: Autor

A vizinhança de um vértice v, denotada por $N(v)$, é o conjunto formado por todos os vértices adjacentes a v. Dois vértices são adjacentes se existe uma aresta entre eles. A cardinalidade de N(v) é chamada de grau de v e é denotado por d(v). O maior grau entre todos os vértices de um grafo por $\Delta(G)$.

2.2 Pesquisa operacional

Problemas de otimização possuem em sua naturalidade, uma função objetivo para maximização ou minimização, seguida de uma série de funções de restrição, impostas as variáveis criadas para simular as decisões ou valores a serem conhecidos.

Otimização combinatória pode ser descrita como o estudo dos problemas de otimização em conjuntos finitos e discretos (CHAVES, 2009). Formulações matemáticas, podem ser ditas como a escrita do problema de otimização, com uma função objetivo seguida por uma série de restrições para o conjunto de variáveis especificado.

2.3 Programação linear e linear inteira

A programação linear é tida como uma das técnicas de pesquisa operacional de maior destaque devido a sua simplicidade de entendimento e geração de modelos matemáticos para sua resolução. Esta, constitui um caso onde as variáveis são continuas e apresentam comportamento linear em relação as restrições impostas a elas, permitindo uma maior eficiência e fácil implementação (SILVA *et al.*, 2019).

Quando todas as variáveis de um modelo de programação linear são inteiras, chamamos esse problema de programação inteira (PLI), podendo ser vista como um problema de

programação linear mais uma restrição de que suas variáveis são inteiras. Caso todas as variáveis de um modelo PLI são inteira, diz-se que o problema é inteiro puro, caso contrário, chama-se o modelo de programação inteira mista (ALMEIDA *et al.*, 2017).

2.4 Heurística e heurística construtiva

O termo heurística é derivado da Grécia, significando um atalho, uma aproximação, uma regra de busca, etc. Heurísticas são estratégias propostas para ignorar parte do conjunto solução de um problema na esperança de obter uma solução de qualidade em tempo curto (CAMPO *et al.*, 2016).

Existem vários tipos de heurísticas, dentre elas se pode citar heurísticas de reconhecimento, heurísticas *take-the-best* e heurísticas construtivas. Heurísticas construtivas constroem uma solução para o problema usando características particulares do problema.

3 O PROBLEMA DE INFECÇÃO DE GRAFOS

Nessa sessão serão abordados os conceitos relacionados a problemática de infecção em grafos além de explicar e exemplificar as regras de infecção $P3$ e $P4$. As regras de infecção $P3$ e $P4$ foram escolhidas para o trabalho por serem as mais fáceis de se trabalhar dentro do problema de infecção de grafos.

Em otimização, um problema de infecção pode ser expresso como o problema de determinar um conjunto $S \in V$ de vértices a serem inicialmente infectados em um grafo $G = (V,E)$ conexo, finito e não direcionado, de modo a espalhar um vírus no grafo G entre os vértices que são suscetíveis a infecção de acordo com uma regra pré-definida (DRAIEF; GANESH, 2010).

No problema de infecção é dada uma função I(S), chamada de função de infecção, esta função retorna o conjunto de vértices que é infectado pelo conjunto S, de acordo com a regra de infecção definida.

O processo de infecção se dá por iterações no tempo. Dado o conjunto de vértices inicialmente infectados S (tempo 0), aplicamos a função de infecção e assim temos o conjunto de vértices infectados no tempo 1, $S \cup i(S)$. O processo continua, infectando no tempo 2 os vértices em $I(S \cup I(S))$ e assim por diante, até que todos os vértices sejam infectados ou não haja mais como infectar um vértice.

3.1 Regra de Infecção P3

A regra de infecção P_3 está associada a existência de caminhos de comprimento 2, ou seja, com 3 vértices. Seja xyz um caminho de comprimento 2, se x e z estão infectados no tempo i, então y é infectado no tempo i+1 (CENTENO, 2012).

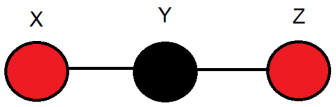

Figura 5 – Exemplo $P3$
Fonte: Autor

3.1.1 *Exemplo de infecção por P3*

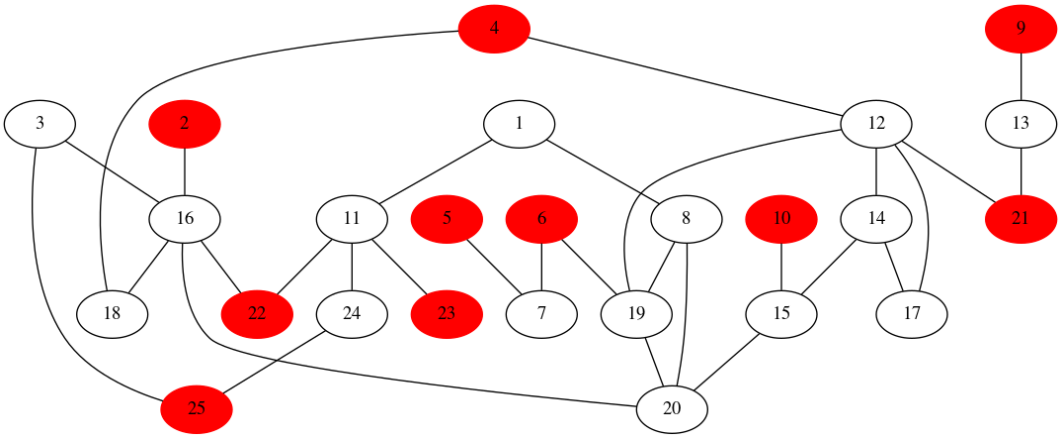

Figura 6 – Grafo infectado por regra P3
Fonte: Autor

A Figura 1 ilustra um exemplo da regra de infecção P_3. O conjunto infectado inicialmente S em um tempo i, pode ser visto nos vértices destacados em vermelho, onde a cada passo de interação, os vértices que respeitarem a regra de infecção P3 serão infectados. Esse passo de infecção é repetido até que todos os vértices do grafo sejam infectados. No passo de infecção $i+1$, os vértices 7, 11, 12, 13, 16 serão infectados por possuírem dois vizinhos infectados no tempo anterior i, assim respeitando a regra. No passo de infecção seguinte, os vértices 3, 18, 19, 24 serão infectados. No próximo passo, apenas o vértice 20 será infectado, seguido a esse passo de iteração, os vértices 15 e 8 serão infectados, no passo seguinte, 1 e 14 são infectados e finalmente, o vértice 17 é infectado, fazendo com que todos os vértices do grafo estejam infectados.

3.2 Regra de Infecção P4

A regra de infecção P_4 está associada a existência de caminhos de comprimento 3, ou seja com 4 vértices. Seja $xyfz$ um caminho de comprimento 3, se x e z estão infectados no tempo i, então y e f são infectado no tempo i+1 (ENRIQUEZ; JUNIOR, 2014).

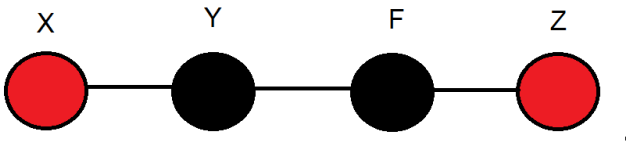

Figura 7 – $P4$ com vértices infectados em vermelho
Fonte: Autor

3.2.1 Exemplo de infecção por P4

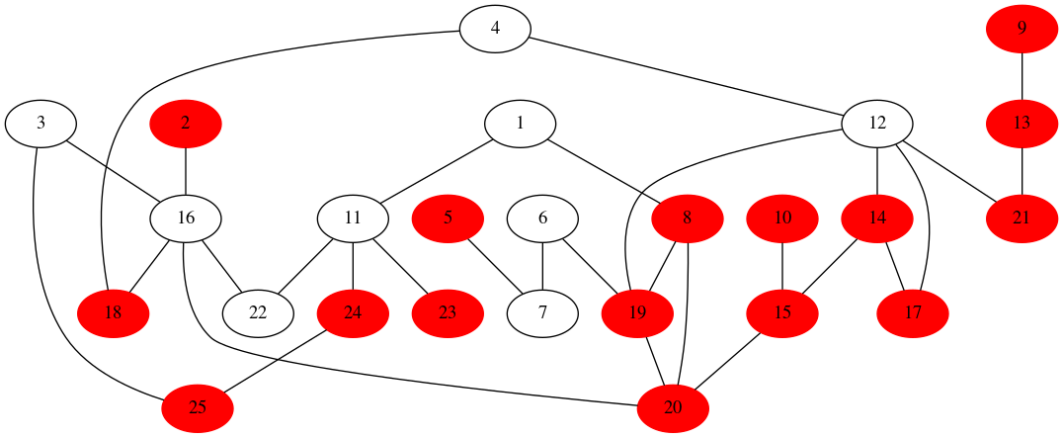

Figura 8 – Grafo infectado por regra P4
Fonte: Autor

Na figura é mostrado um exemplo da regra de infecção P_4, onde os vértices destacados em vermelho como sendo o conjunto inicial de infectados S do grafo G dado. Onde a cada iteração, os vértices que foram suscetíveis a infecção respeitando a regra de infecção P4 serão infectados. No passo de interação $i+1$, todos os vértices são infectados, pois todos respeitam a regra de infecção $P4$.

4 TRABALHOS RELACIONADOS

Nesta sessão será apresentado a literatura para a produção e escrita do trabalho. Vale ressaltar que não existem muitos trabalhos no âmbito de otimização envolvendo infecção de grafos, por se tratar de um problema mais recente.

4.1 Trabalhos em teoria dos grafos

O tratamento desse tema sobre o prisma de otimização combinatória é bem recente, até o melhor do conhecimento, não existem muitos trabalhos nessa área sobre esse assunto. Entretanto, existem diversos trabalhos em teoria dos grafos sobre esse conteúdo. O artigo Dimotriou *et al.* (2006) traz uma ideia de como é a resolução do problema por termos matemáticos, utilizando limites superiores, servindo como base para provar que o problema é viável.

O artigo, Dimotriou *et al.* (2006), busca resolver o problema de infecção de grafos, para k partículas em um movimento de infecção aleatório ao longo do tempo, permitindo que um vértice infectado transmita essa infecção a outro vértice saudável apenas ao encontrá-lo a partir do movimento aleatório pelo grafo.

Draief e Ganesh (2010) trazem uma prova matemática para a solução do problema de infecção usado uma regra de infecção, levando em conta a popularidade dos vértices do grafo para a visita do vizinho infectado. Esse trabalho, aborda temas importantes para a prova matemática utilizando conceitos de álgebra linear, além proporcionar uma destacada fundamentação sobre o problema.

Sobre o ponto de vista de otimização, existem poucos trabalhos sobre infecção em grafos, a seguir são apresentados os dois mais relevantes.

4.2 Uma formulação inteira para o problema *The Target Set Selection*

O trabalho aborda o problema *Target Set Selecion*, onde tem-se uma característica que se espalha, como uma infecção no grafo, a partir de um conjunto S de infectados, em um grafo $G = (V, E)$. Cada vértice tem uma resistência à essa infecção. Inicia-se o processo de infecção infectando os vértices iniciais e se espalhando para um novo vértice se já tiver vizinhos suficientemente infectados. O objetivo do trabalho é encontrar o menor conjunto inicial de vértices infectados que espalhe sua infecção por todo o grafo (PANIZZI *et al.*, 2019).

4.3 Calculando o Número de Envoltória nas Convexidades $P3$ e $P3^*$

O trabalho aborda a envoltória convexa de um conjunto de vértices S. Um subconjunto de S em um grafo $G = (V,E)$ é convexo na convexidade $P3$ se todo vértice $u \in V(G)$ não possuir dois vizinhos em S. A envoltória convexa de S, pode ser dita como o menor conjunto convexo onde S está contido. No trabalho, são propostos formulações de programação linear inteira para determinar o número de envoltória de um grafo utilizando as convexidades $P3$ e $P3^*$, onde na primeira é retornado pela função de infecção o conjuntos dos vértices com dois vizinhos do grafo G, e na segunda, é retornado os vértices que possuem dois u, v como vizinhos. O documento também, realiza testes computacionais para avaliar o desempenhos das formulações obtidas (ARAUJO *et al.*, 2018).

5 CONTRIBUIÇÕES

Na seção de contribuições, são apresentados tudo o que foi desenvolvido para se puder resolver o problema de infecção em grafos, contando um duas formulações matemáticas para regras de infecção $P3$ e $P4$ além de uma heurística construtiva implementada.

Após o estudo do problema e o do referencial teórico, foi possível o desenvolvimento das formulações matemáticas para as regras de infecção $P3$ e $P4$, que diferem das formulações apresentadas na literatura, onde a formulação para a regra de infecção $P4$ pode ser generalizada para regras de infecção posteriores, como $P5, P6, P7$, etc., formulações $P3$ já aparecem na literatura mais recente Araujo *et al.* (2018).

Com a formulação dos modelos para as regras de infecção, foi então realizada a implementação dos algoritmos exatos para as regras $P3$ e $P4$. Foi utilizada a biblioteca PuLP da linguagem Python juntamente com o solver CPLEX para implementação e teste dos modelos.

Além disso, iniciou-se o processo de implementação de uma heurística para o problema de infecção utilizando cada uma das regras. Para tanto, foi então desenvolvida uma heurística construtiva, que também foi implementada utilizado Python como linguagem para implementação.

A heurística desenvolvida se baseia no maior grau do grafo, como a métrica principal de seleção para ser inserido no conjunto de infectados. Optamos por essa métrica pois foi percebido a partir de teste realizados utilizando os algoritmos exatos, que na maioria dos resultados obtidos, os vértices de maior grau faziam parte da solução.

Nas próximas seções deste trabalho serão abordados as contribuições alcançadas, começando com as formulações matemáticas para as regras de infecção $P3$ e $P4$ e em seguida serão mostradas as heurísticas obtidas para ambas as regras de infecção para resolução do problema de infecção.

5.1 Formulação Matemática

Para modelar o problema de infeção, foram criados dois conjuntos de variáveis binárias, um chamado de x_{vt}, que é utilizada tanto para $P3$ quanto $P4$ e y_{wvu}, que é utilizada para a infecção $P4$. A variável binária x_{vt}, no qual o v diz respeito a um vértice e t representa uma iteração (tempo) do processo de infecção do grafo, esta variável será 1 quando o vértice v foi infectado no tempo t e 0, caso contrário. A variável y_{wvu} terá um valor $y_{wvu} = 1$ quando existir

um $P4$ entre w e u na qual v esta interligado a w.

5.1.1 Regra de Infecção $P3$

Para infecção $P3$ pode-se escrever o seguinte modelo para o problema de infeção sobre a regra $P3$:

$$min \sum_{v \in V} x_{v0} \tag{5.1}$$

s.a.
$$2x_{vt} \leq \sum_{u \in N(v)} x_{ut-1} + 2x_{vt-1} \qquad \forall v \in V, \forall vu \in E \tag{5.2}$$

$$x_{vt} \geq x_{vt-1} \tag{5.3}$$

$$\sum_{v \in V} x_{vn} \geq |V| \tag{5.4}$$

Modelo 1: Modelo matemático com regra de infecção $P3$

O modelo acima, tem por função objetivo (5.1), que minimiza a quantidade de vértices infectados do um grafo $G = (V, E)$, inicialmente. A restrição (5.2), representa que para todo vértice infectado v é necessário que vizinhos que foram infectados em um tempo $t-1$ estejam ligados a ele. A restrição (5.3), implica que um vértice infectado continua infectado. A última restrição (5.4) corresponde ao somatório de todos os vértices infectados, que tem que ser maior ou igual ao número total de vértices do grafos, implicando que todos os vértices do grafo devem ser infectados.

5.1.2 Regra de Infecção $P4$

A seguir apresentamos um modelo de programação matemática para o problema de infecção com a regra de infecção $P4$.

$$min \sum_{v \in V} x_{v0} \tag{5.5}$$

s.a. $$x_{vt} - \sum y_{uwt-1} - x_{vt-1} \leq 0 \qquad \forall v, \exists wv, vu, us \in E, \exists w \in N(u), u \in N(v) \tag{5.6}$$

$$x_{vt} \geq x_{vt-1} \tag{5.7}$$

$$y_{wut-1} \leq x_{wt-1} \tag{5.8}$$

$$y_{wut-1} \leq x_{ut-1} \tag{5.9}$$

$$\sum_{v \in V} x_{vn} \geq |V| \tag{5.10}$$

Modelo 2: Modelo matemático com regra de infecção $P4$

O modelo acima, basicamente possui a mesma formulação matemática do modelo anterior, onde tem por função (5.5) minimizar a quantidade de vértices infectados do um grafo $G = (V, E)$ dado. A restrição (5.6) do modelo apresentado, representa que para todo vértice infectado v_{vt} é necessário haver um vizinho infectado e um vizinho que possua uma ligação com outro vértice que também esta infectado, ambos infectados em um tempo $t-1$ estejam ligados a v, satisfazendo a restrição. A segunda restrição (5.7), implica que um vértice infectado não deixa de ser infectado.As restrições (5.8) e (5.9) dizem que o par ordenado da aresta deve estar infectado em um tempo anterior. A última restrição (5.10) corresponde ao somatório de todos os vértices infectados, que tem que ser maior ou igual ao número total de vértices do grafos, implicando que todos os vértices do grafo devem ser infectados.

5.2 Heurística

Na heurística construtiva implementada, foi utilizado um passo inicial em que, se um vértice $v \in V$ possua grau 1, $(\dot{v}) = 1$, então automaticamente este vértice já é inserido no conjunto de vértices inicialmente infectados que é retornado no final como solução, já que como não possuem dois vizinhos, esses vértices não conseguiriam ser infectados durante o processo de infecção do grafo para as regras utilizadas. A partir disso a heurística irá selecionar um vértice v ainda não infectado, com maior grau dentre os não infectados do grafo para que esse seja infectado, utiliza-se este critério na esperança de que, se vértice selecionado possui grau elevado, alcança mais vizinhos, logo, consegue infectar mais vértices. Esse processo é repetido até que todos os vértices do grafo possam ser infectados pelo conjunto de vértices escolhido. Ao final do algoritmo, o programa realiza uma verificação, essa verificação consiste em analisar se

um vértice dentre os escolhidos para o conjunto de vértices inicialmente infectados pode ser infectado pelo conjunto de vértices escolhidos menos ele. Se sim, então este vértice é removido do conjunto.

5.2.1 Algoritmo da Heurística Construtiva

Algoritmo 1: Algoritmo Heurístico

Input: Um Grafo G conectado

Output: Lista L dos primeiros vértices infectados do Grafo G

```
1  criar lista vazia S
2  criar lista vazia L
3  P ← V(G)
4  S = S ∪ vértices com grau 1 de G
5  P = P− vértices com grau 1 de G
6  while P ≠ ∅ do
7      Selecionar um vértice v de maior grau de P
8      Adiciona v em S
9      Retira v em P
10     P ← P\ infecta(G, S)
11     S ← S∪ infecta(G, S)
12 while S ≠ ∅ do
13     Selecionar um vértice v de S sequencialmente
14     if se v não é infectado por S-v then
15         L = v∪L
16     Retira v de S
```

Acima é demonstrado um exemplo do funcionamento da heurística construtiva utilizada no problema de infecção. Onde *infecta* é uma função que retorna os vértices que são infectados no próximo passo de interação com a inserção de v ao conjunto S, podendo ser diferente quando aplicada a regra de infecção $P3$ e $P4$. O algoritmo heurístico para regra $P3$ possui complexidade $O(n^4)$ pois no pior caso, irá precisar percorrer todos os seus laços de repetição para encontrar a solução. Para caso da regra $P4$ o algoritmo irá ter complexidade $O(n^5)$.

5.2.2 Regra de Infecção P3

Algoritmo 2: Algoritmo infectaP3

Input: Um Grafo G conectado, conjunto S de infectados de G

Output: Lista Q dos vértices infectados em um passo de infecção

```
1  foreach v ∈ S do
2      foreach u ∈ S, u ≠ v do
3          for m ∈ N(v) do
4              if m ∈ N(j) then
5                  if m ∉ Q then
6                      Q = Q ∪ m;
7  return Q;
```

O algoritmo infectaP3 faz a disseminação da infecção a partir do conjunto de infectados S. O algoritmo verifica quais os vértices serão infectados por possuírem vizinhos infectados em S respeitando a regra de infecção $P3$.

5.2.3 Regra de Infecção P4

Algoritmo 3: Algoritmo infectaP4

Input: Um Grafo G conectado, conjunto S de infectados de G

Output: Lista S dos vértices infectados em um passo de infecção

```
1  foreach v ∈ S do
2      foreach u ∈ S, u ≠ v do
3          for w ∈ S, v ≠ v, u do
4              if v não esta infectado then
5                  if u esta infectado e u ∈ N(v) then
6                      if w não esta infectado e w ∈ N(v) then
7                          foreach m ∈ N(w), m ≠ u do
8                              if m não esta em infectados then
9                                  Q = Q ∪ m;
10 return Q;
```

O algoritmo infectaP4 faz a disseminação da infecção a partir do conjunto de infecta-

dos S. O algoritmo verifica quais os vértices serão infectados por possuírem vizinhos infectados em S respeitando a regra de infecção $P4$.

6 RESULTADOS

Os resultados a seguir, foram obtidos a partir da utilização do algoritmo exato e heurístico utilizando regra de infecção $P3$. A regra de infecção $P4$ não foi testada por questões de tempo.

6.1 Algoritmo exato

A seguir serão apresentados os dados obtidos a partir do teste nos algoritmos exatos para o problema de infecção utilizando regra de infecção P3, para dados mais completos, verificar ANEXO A ao final do documento. Foi desenvolvido dois conjuntos de instâncias randômicas: puras e conexas, onde as puras, todas as arestas do grafo são gerados aleatoriamente, podendo haver mais de uma aresta entre dois vértices ou ainda, vértices de grau 0, já os conexos, devem possuir apenas uma aresta entre dois vértices além de não permitir vértices de gray 0. Para ambos os conjuntos, definiu-se então, um tamanho para o conjunto de vértices (5,10,15,20,25,30 e 35) e uma densidade para o grafo (20%, 30%, 50%, 70%, 90%). A densidade de um grafo é a razão entre o número de arestas que este possui e o número de arestas de uma clique com os mesmos vértices. As arestas foram geradas de maneira randômica, com a diferença que para o conjunto de instâncias conexo randômico, é exigido que o grafo seja conexo.

A partir da obtenção das instâncias, foi feito um processo de teste. A formulação foi executada 5 vezes. O tempo reportado é a média do tempo de execução para essas 5 execuções. Isto é feito para evitar que variações de memória ou de cache interfiram nos resultados. A seguir serão mostrados as figuras de resultados para grafos conexos e grafos randômicos onde são apresentados as soluções encontradas pelos algoritmos como resultados.

6.2 Algoritmo heurístico

Foram utilizadas as mesmas instâncias para testar o algoritmo heurístico proposto. Foi realizado 10 execuções, pois como este algoritmo é combinatório e esta mais propenso a interferências oriundas do escalonamento dos processos na CPU e da utilização da memória, pode ser que algum processo da máquina cause influência na execução do algoritmo, por isso é necessário uma maior quantidade de testes e sua média para evitar valores não condizentes com o real, juntamente com a realização da média dos tempos de execução. A seguir serão mostrados as figuras de resultados para grafos conexos e grafos randômicos:

Pode-se observar que a heurística ainda apresenta um comportamento que parece ser exponencial. Esse comportamento exponencial do algoritmo heurístico, pode ser explicado com duas possibilidades, primeiro, pode ser que o tamanho dos grafos não permita que seja observado o comportamento polinomial. Outra razão pode ser a linguagem de implementação escolhida que pode ser lenta na manipulação de listas.

6.3 Comparando os resultados

A partir dos dados obtidos anteriormente, foi realizado um comparativo entre o algoritmo exato $P3$ e a heurística $P3$. As figuras a seguir, foram produzidas a partir da comparação de todas as soluções de todos os grafos formados para com quantidade de arestas de 20%, 30%, 500%, 70% e 80%. Os resultados foram obtidos a partir dos resultados obtidos das tabelas listadas no anexo 1 e foram plotados utilizando a biblioteca *matplotlib* do *Python*.

6.3.1 Comparativo entra randômicos conexos exato e heurístico

As figuras a seguir, são uma comparação dos dados gerados pelas soluções obtidas a partir do algoritmo exato e heurístico aplicados a grafos conexos.

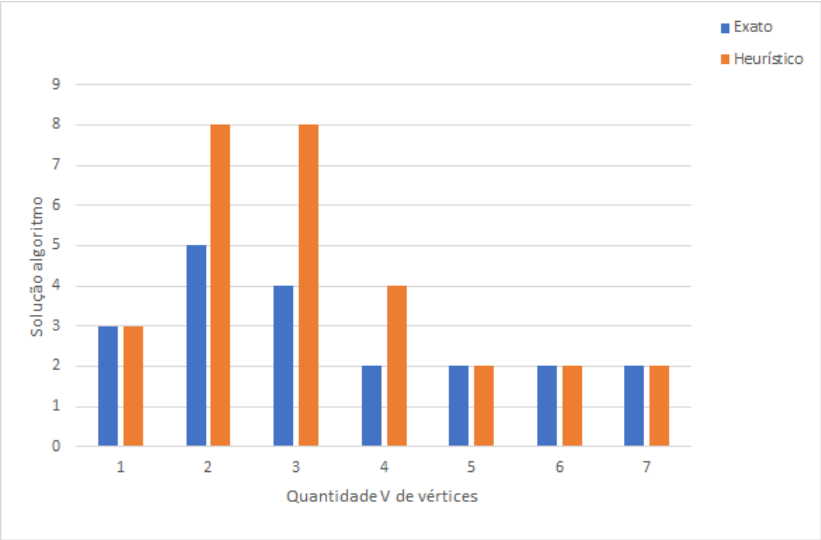

Figura 9 – Comparativo de solução para 20% arestas

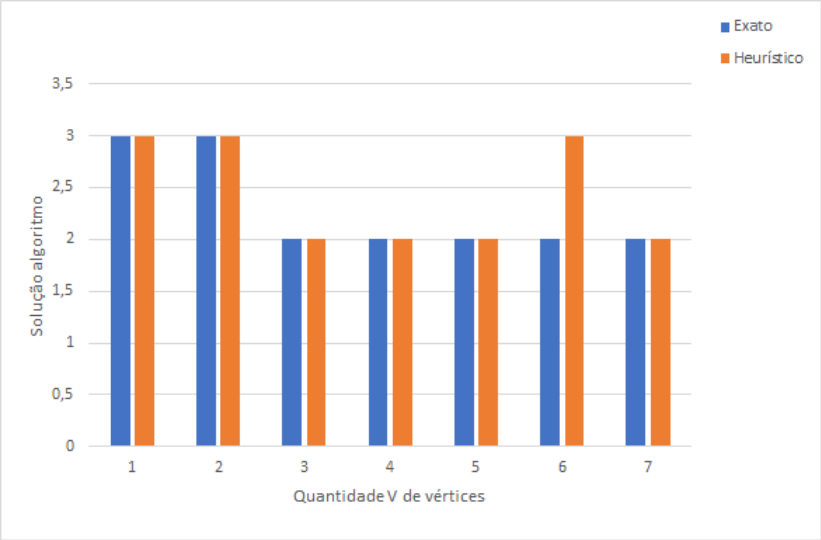

Figura 10 – Comparativo de solução para 30% arestas

Para grafos com densidade inferior a 30% de arestas, o algoritmo heurístico encontra o resultado mais rápido que o exato, mas as soluções possuem GAP muito distante do valor exato.

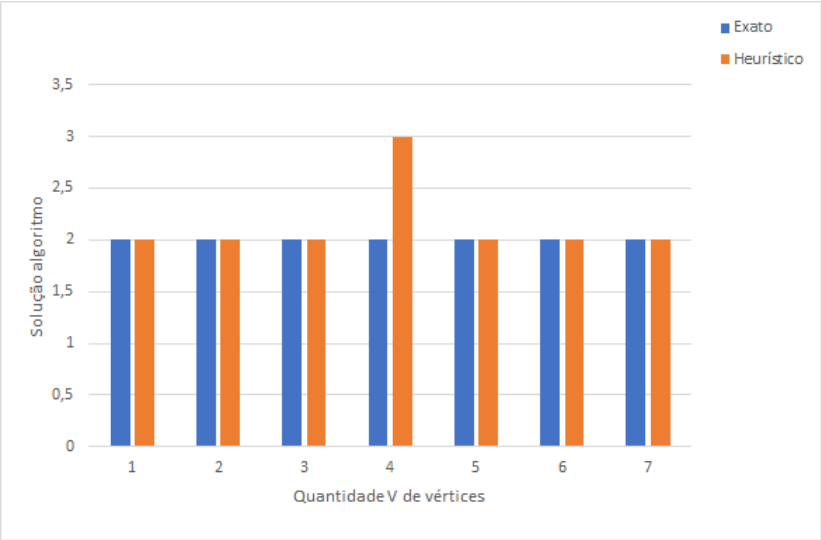

Figura 11 – Comparativo de solução para 50% arestas

A partir de 50% de arestas, o algoritmo heurístico começa a estabilizar os resultados encontrado, chegando a ter mais valores perto ou igual ao exato.

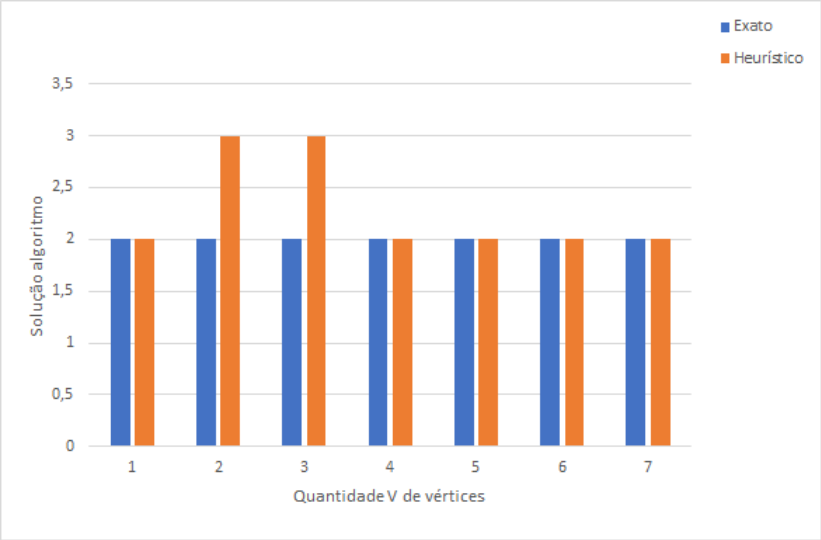

Figura 12 – Comparativo de solução para 70% arestas

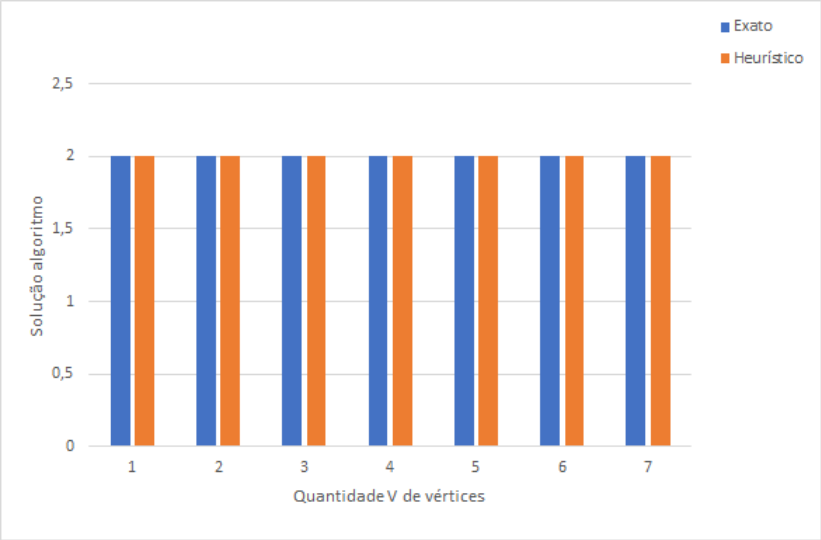

Figura 13 – Comparativo de solução para 80% arestas

Com mais densos com 80% de arestas, o algoritmo heurístico se sai muito bem, encontrando soluções iguais as soluções exatas e com menor tempo de execução.

6.3.2 Comparativo entra randômicos puros exato e heurístico

As figuras a seguir, são uma comparação dos dados gerados pelas soluções obtidos a partir do algoritmo exato e heurístico aplicados a grafos randômicos.

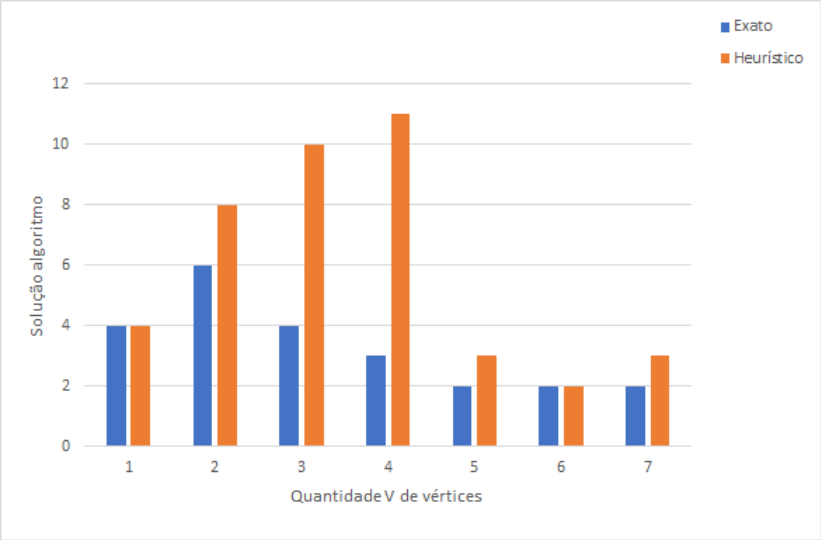

Figura 14 – Comparativo de solução para 20% arestas

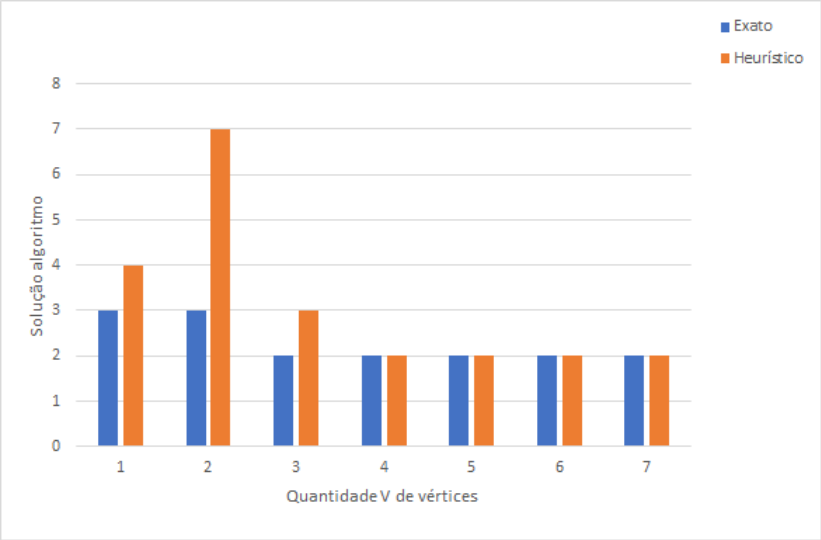

Figura 15 – Comparativo de solução para 30% arestas

Como ocorre com grafos randômicos conexos, para densidades de arestas menores ou iguais a 30%, o algoritmo heurístico encontra a solução mais rápido, mas com GAP em muitos casos, distante da solução exata.

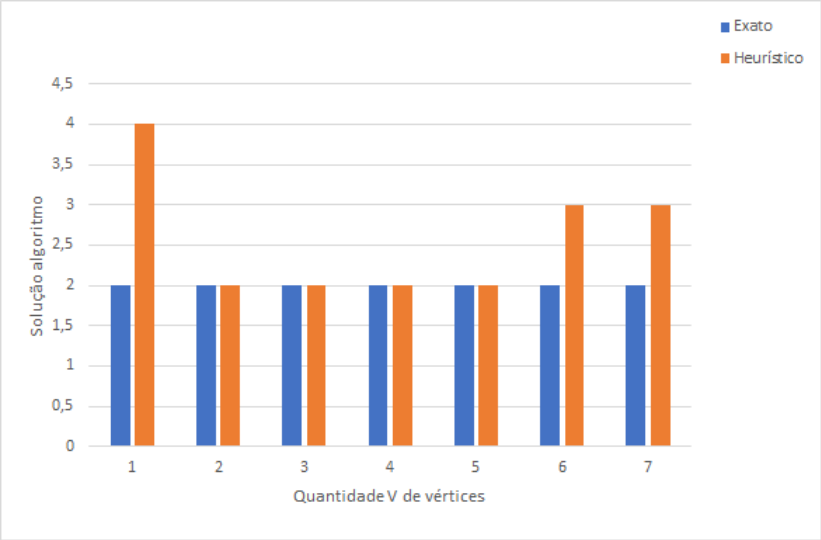

Figura 16 – Comparativo de solução para 50% arestas

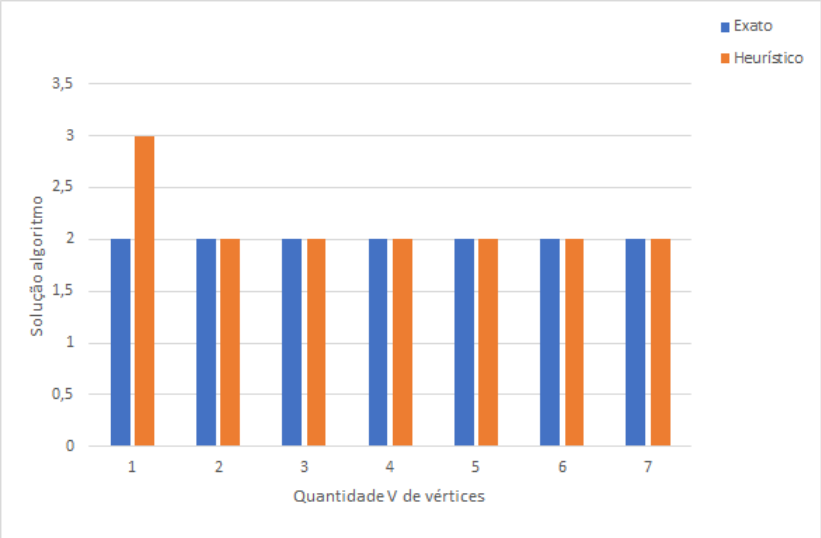

Figura 17 – Comparativo de solução para 70% arestas

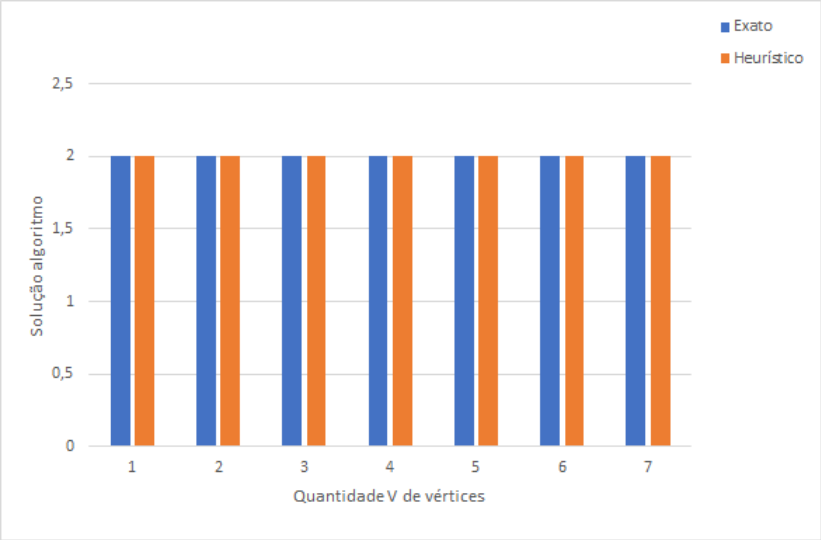

Figura 18 – Comparativo de solução para 80% arestas

Para grafos randômico puros com densidades grandes com 80% de arestas, o algoritmo heurístico irá encontrar a solução mais rápida que o exato, possuindo a mesma solução do mesmo.

6.4 GAP

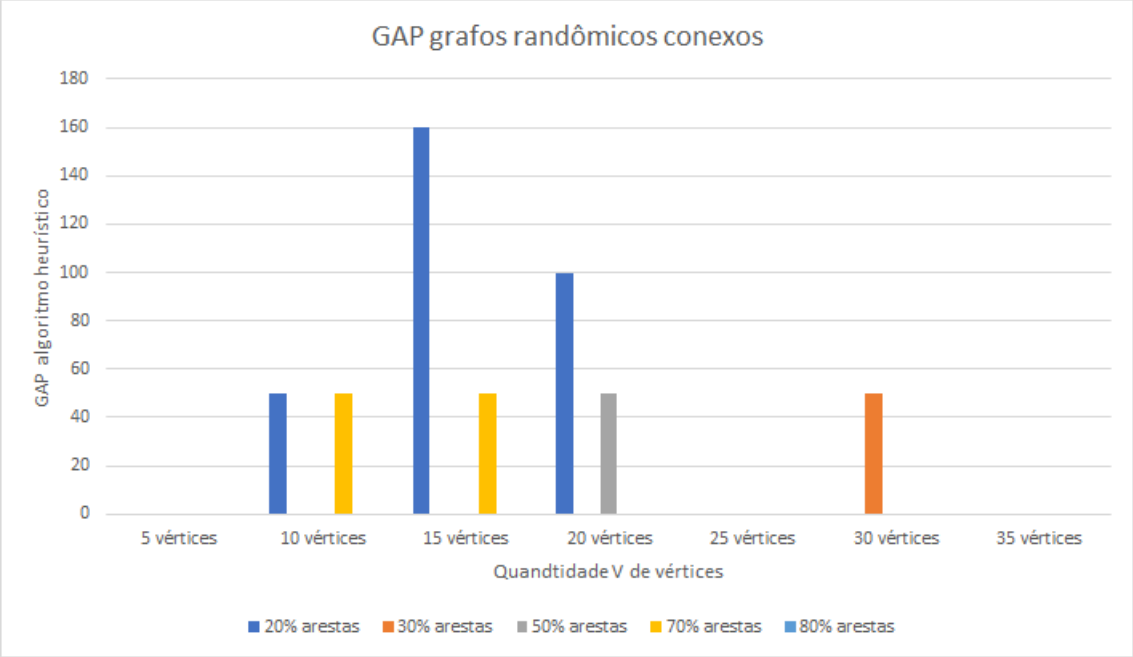

Figura 19 – GAP com todas as densidades de arestas para grafos randômicos conexos

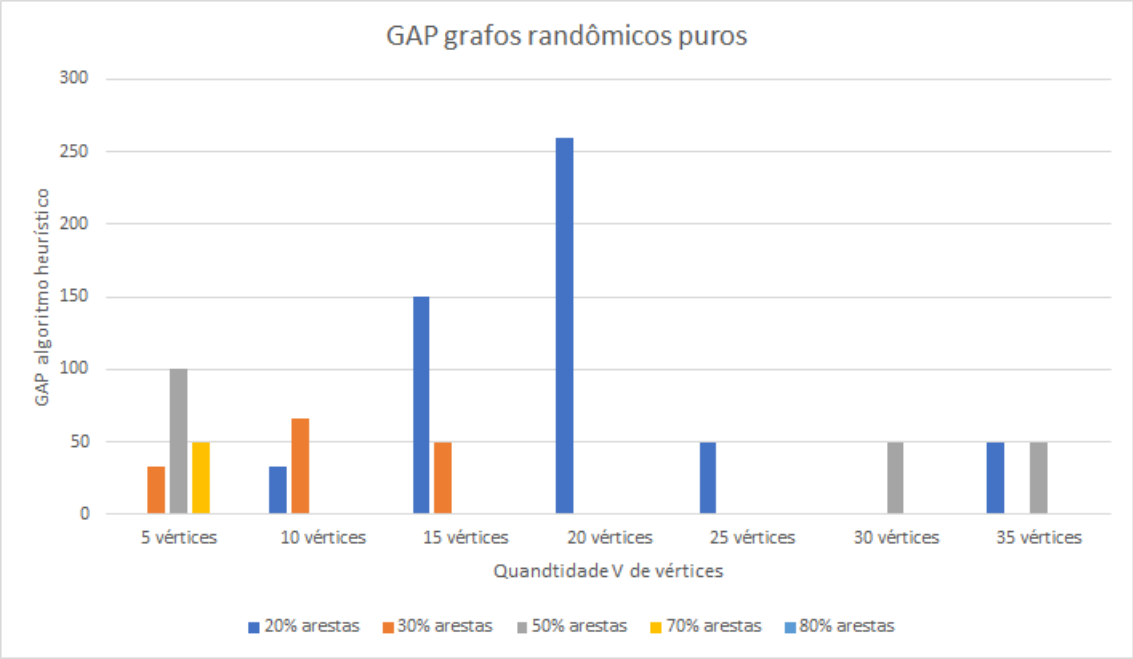

Figura 20 – GAP com todas as densidades de arestas para grafos randômicos puros

Observe que a heurística ainda é muito mais rápida que o algoritmo exato e que o GAP (distância relativa da solução ótima) é pequeno para grafos com densidade mais altas.

Com a análise dos resultados a partir dos dados obtidos, foi possível verificar que a heurística *P*3 implementada, consegue vencer em muitos casos o tempo de execução do algoritmo exato, muitas vezes conseguindo até, o mesmo valor de solução encontrado pelo algoritmo exato, sendo que para valores abaixo de 50% de arestas, a heurística tenha dificuldade de achar a solução mais próxima do exato em alguns casos com quantidades abaixo de 20 vértices.

7 CONCLUSÃO

Com a validação dos dados, também foi perceptivo que o algoritmo heurístico é satisfatório para solucionar o problema para instâncias com densidade maior que 50%, sendo que em muitos casos, a distância da solução para a solução exata foi nula ou quase nula, só em alguns casos o GAP teve valores distantes em comparação ao exato.

Foi percebido também, que a utilização da heurística em grafos conexos gera valores superiores em comparação aos dados obtidos utilizando grafos randômicos, já que o resultado final dos conexos, fica bem mais próximo do exato do que os grafos randômicos. Também foi percebido que para a heurística se sai muito bem para grafos com densidades altas mas que não possui um bom comportamento para baixas densidades. Por último, foi identificado que o algoritmo exato demora mais tempo resolvendo instâncias com médias densidades, resolvendo pequenos e grandes problemas rapidamente.

Para trabalhos futuros, será implementada a heurística para a regra de infecção $P4$, para que sendo esta, testada e validada, seja possível comprovar que a generalização para regras maiores, também vá funcionar, já que a partir de $P4$, pode-se chegar a outras regras de infecção.

REFERÊNCIAS

ALMEIDA, W. S. de; SILVA, M. E. da; SANTOS, N. V. M. dos; RIBEIRO, L. M.; BACHEGA, S. J. **Aplicação de programação linear inteira na maximização do lucro de uma empresa do setor de beleza e estética**. 2017.

ARAUJO, J.; CAMPêLO, M.; SOUSA, G. H. de. **Calculando o número de envoltória nas convexidades P3 e P3** †. In: **Anais do III Encontro de Teoria da Computação**. Porto Alegre, RS, Brasil: SBC, 2018. ISSN 2595-6116.

CAMPO, C. del; PAUSER, S.; STEINER, E.; VETSCHERA, R. **Decision making styles and the use of heuristics in decision making**. **Journal of Business Economics**, v. 86, n. 4, p. 389–412, May 2016. ISSN 1861-8928.

CENTENO, C. C. **A Convexidade P3 para Grafos não Direcionados**. 2012.

CHAVES, A. A. **Meta-heurística híbrida com busca por agrupamentos aplicada a problemas de otimização combinatória**. 2009.

DIMOTRIOU, T.; NIKOLETSEAS, S.; SPIRAKIS, P. **The infection time of graphs**. Elsevier BV, 2006.

DRAIEF, M.; GANESH, A. **A random walk model for infection on graphs: spread of epidemics and rumours with mobile agents**. Springer Nature, 2010.

ENRIQUEZ, E. L.; JUNIOR, S. R. C. **A short note on convexity and convex domination in G[Km]**. Hikari, Ltd, 2014.

Hosseinalipour, S.; Wang, J.; Tian, Y.; Dai, H. **Infection Analysis on Irregular Networks through Graph Signal Processing**. **arXiv e-prints**, p. arXiv:1808.04879, Aug 2018.

PANIZZI, A.; SANTOS, V. F. dos; URRUTIA, S. an. **An Integer Formulation For The Target Set Selection Problem**. 2019.

SANTOS, M. **CICLOS HAMILTONIANOS EM GRAFOS**. **Ciência e Natura**, v. 39, n. 3, 2017.

SEIBOLD, C.; CALLENDER, H. L. **Modeling epidemics on a regular tree graph**, journal = Letters in Biomathematics. Taylor Francis, v. 3, n. 1, p. 59–74, 2016.

SILVA, R. R. da; SOARES, C. M. da S.; AGUIAR, A. O. de; GOMES, D. da S.; MARTINS, G. A. de S.; SILVA, W. G. da. **Uso da programação linear na otimização de processos dentro da industria de alimentos**. 2019.

SOUSA, H. **Um estudo sobre cliques-dominantes em grafos**. 2015.

ANEXO A – TABELAS DE DADOS OBTIDOS

A.1 Algoritmo exato

Grafo P3 conexo com 20% de arestas			
tam(V)	SOL	GAP	TEMPO
5	3	0	0.139
10	5	0	0.309
15	3	0	1.414
20	2	0	2.840
25	2	0	6.155
30	2	0	13.926
35	2	0	23.762

Grafo P3 randômico com 20% de arestas			
tam(V)	SOL	GAP	TEMPO
5	4	0	0.552
10	6	0	0.735
15	4	0	2.826
20	3	0	4.976
25	2	0	12.522
30	2	0	38.171
35	2	0	68.958

Tabela 1 – Grafo com 20% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 30% de arestas			
tam(V)	SOL	GAP	TEMPO
5	3	0	0.129
10	3	0	0.342
15	2	0	1.322
20	2	0	3.785
25	2	0	7.712
30	2	0	14.838
35	2	0	25.420

Grafo P3 randômico com 30% de arestas			
tam(V)	SOL	GAP	TEMPO
5	3	0	0.291
10	3	0	0.713
15	2	0	1.332
20	2	0	3.236
25	2	0	7.905
30	2	0	16.340
35	2	0	32.316

Tabela 2 – Grafo com 30% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 50% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.160
10	2	0	0.432
15	2	0	2.125
20	2	0	4.774
25	2	0	9.239
30	2	0	63.630
35	2	0	145.528

Grafo P3 randômico com 50% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.627
10	2	0	0.893
15	2	0	4.188
20	2	0	7.725
25	2	0	14.330
30	2	0	81.745
35	2	0	98.691

Tabela 3 – Grafo com 50% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 70% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.144
10	2	0	0.498
15	2	0	2.497
20	2	0	9.364
25	2	0	14.603
30	2	0	41.965
35	2	0	64.946

Grafo P3 randômico com 70% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	1.249
10	2	0	1.762
15	2	0	4.560
20	2	0	14.943
25	2	0	45.167
30	2	0	99.591
35	2	0	121.566

Tabela 4 – Grafo com 70% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 80% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.356
10	2	0	0.666
15	2	0	2.487
20	2	0	8.883
25	2	0	32.628
30	2	0	63.601
35	2	0	118.532

Grafo P3 randômico com 80% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.146
10	2	0	0.484
15	2	0	2.773
20	2	0	10.743
25	2	0	26.996
30	2	0	55.325
35	2	0	109.401

Tabela 5 – Grafo com 70% de arestas resolvidos pela regra de infecção P3

A.2 Algoritmo heurístico

Grafo P3 conexo com 20% de arestas			
tam(V)	SOL	GAP	TEMPO
5	3	0	0.210
10	8	50%	0.373
15	8	160%	0.607
20	4	100%	0.957
25	2	0	1.683
30	2	0	4.433
35	2	0	5.946

Grafo P3 randômico com 20% de arestas			
tam(V)	SOL	GAP	TEMPO
5	4	0	0.121
10	8	33%	0.224
15	10	150%	0.975
20	11	260%	9.182
25	3	50%	9.593
30	2	0	10.374
35	3	50%	11.339

Tabela 6 – Grafo com 20% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 30% de arestas			
tam(V)	SOL	GAP	TEMPO
5	3	0	0.156
10	3	0	0.276
15	2	0	0.498
20	2	0	1.030
25	2	0	2.079
30	3	50%	3.631
35	2	0	7.102

Grafo P3 randômico com 30% de arestas			
tam(V)	SOL	GAP	TEMPO
5	4	33%	0.088
10	7	66%	0.306
15	3	50%	0.574
20	2	0	1.208
25	2	0	2.455
30	2	0	4.042
35	2	0	7.651

Tabela 7 – Grafo com 30% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 50% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.211
10	2	0	0.451
15	2	0	0.916
20	3	50%	1.838
25	2	0	3.729
30	2	0	8.449
35	2	0	18.011

Grafo P3 randômico com 50% de arestas			
tam(V)	SOL	GAP	TEMPO
5	4	100%	0.138
10	2	0	0.388
15	2	0	1.164
20	2	0	1.955
25	2	0	4.007
30	3	50%	7.361
35	3	50%	15.415

Tabela 8 – Grafo com 50% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 70% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.157
10	3	50%	0.376
15	3	50%	1.004
20	2	0	2.545
25	2	0	5.274
30	2	0	12.962
35	2	0	30.059

Grafo P3 randômico com 70% de arestas			
tam(V)	SOL	GAP	TEMPO
5	3	50%	0.473
10	2	0	0.739
15	2	0	1.195
20	2	0	2.511
25	2	0	6.399
30	2	0	15.702
35	2	0	33.053

Tabela 9 – Grafo com 50% de arestas resolvidos pela regra de infecção P3

Grafo P3 conexo com 80% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.151
10	2	0	0.566
15	2	0	1.323
20	2	0	3.212
25	2	0	7.407
30	2	0	17.932
35	2	0	38.135

Grafo P3 randômico com 80% de arestas			
tam(V)	SOL	GAP	TEMPO
5	2	0	0.176
10	2	0	0.467
15	2	0	1.091
20	2	0	2.380
25	2	0	5.506
30	2	0	12.931
35	2	0	33.381

Tabela 10 – Grafo com 50% de arestas resolvidos pela regra de infecção P3