Modelagem e Soluções Heurísticas para o Problema de Conjunto Convergente Mínimo

Alana Rasador Panizzi
Universidade Federal de Minas Gerais - UFMG
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901, Brasil
panizzis@gmail.com

Sebastián Urrutia
Universidade Federal de Minas Gerais - UFMG
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901, Brasil
surrutia@dcc.ufmg.br

Vinícius Fernandes dos Santos
Universidade Federal de Minas Gerais - UFMG
Av. Pres. Antônio Carlos, 6627 - Pampulha, Belo Horizonte - MG, 31270-901, Brasil
viniciussantos@dcc.ufmg.br

RESUMO

Processos de conversão irreversível simulam a disseminação de uma certa característica (infecção) em um grafo. Cada vértice possui uma certa tolerância à infecção e, caso o número de vizinhos infectados seja maior ou igual a essa tolerância, este também é infectado. Este artigo é focado no problema de conjunto convergente mínimo de vértices em um grafo, i.e., o conjunto de cardinalidade mínima de infecções iniciais que acarretarão a infecção total do grafo após um certo número de iterações do processo. Propõe-se aqui um modelo de programação inteira para este problema. Além disso, por este ser NP-difícil, exploram-se duas heurísticas construtivas que aproximem o valor da solução, e as compara com as soluções obtidas por um resolvedor a partir do modelo inteiro. Vemos que as duas heurísticas são melhores na resolução das instâncias aqui propostas quando comparadas com o resolvedor, tanto em tempo, quanto em qualidade da solução.

PALAVRAS CHAVE. Processos de Conversão Irreversível. Conjunto Convergente Mínimo. Infecção em Grafos. Heurística.

ABSTRACT

Irreversible conversion processes simulate the spread of a certain characteristic (infection) through a graph. Each vertex has a certain tolerance to the infection, and if the number of infected neighbors is greater or equal to this tolerance threshold, it will also be infected. This article is focused on the minimum convergent vertex set of a graph, i.e., the set of minimum cardinality of initial infections that will lead to the total infection of the graph, after a certain number of process iterations. Here, an integer programming model of this problem is proposed. In addition, since this problem is NP-hard, we explore constructive heuristics that approximate the value of the solution, and compare them with the solutions obtained by an integer programming solver. We see that the two heuristics are better at solving the instances proposed here when compared to the solver, both in time and quality of the solution.

KEYWORDS. Irreversible Conversion Processes. Minimum Convergent Set. Graph Infection. Heuristic.

1. Introdução

Propagação de doenças, de opinião através de redes sociais, ou até mesmo de falhas através de redes distribuídas: Estes são alguns exemplos de Processos de Conversão Irreversível (PCI), que simulam a disseminação de uma certa característica em um subconjunto dos vértices de um grafo, fazendo com que estes passem, por sua vez, tal característica para os demais vértices.

Um processo no qual, após um certo tempo t_0, todos os vértices estão infectados, será chamado de PCI convergente. Neste artigo, estaremos interessados principalmente no problema de Conjunto Convergente Mínimo (CCM), que visa encontrar o subconjunto de vértices de um grafo de menor cardinalidade, tal que se os vértices do conjunto forem infectados inicialmente, eles acarretarão na infecção dos demais após certo período de tempo. Dreyer e Roberts [2009] caracteriza processos irreversíveis *k-threshold*: Cada vértice é infectado se pelo menos k de seus vizinhos já estiverem infectados. O problema CCM para este tipo de processo foi mostrado ser um problema NP-completo para qualquer k maior que 2. Em sequência, em Centeno et al. [2011], mostra-se a NP-completude do problema para $k = 2$, respondendo a conjectura em aberto proposta por Dreyer e Roberts. Centeno et al. generalizaram o problema acima citado, fazendo com que o número de vizinhos infectados necessário para a infecção de um vértice seja modelado por uma função qualquer $f : V \to \mathbb{Z}^+$, denotado por $irr_f(G)$. Neste artigo, também são obtidos resultados que descrevem um princípio de redução geral para $irr_f(G)$, o que leva a algoritmos eficientes para grafos com blocos de estrutura simples, como árvores e grafos cordais. No entanto, em Reddy et al. [2010], por exemplo, vemos que para a classe de grafos bipartidos, o problema é NP-completo,

No artigo Chopin et al. [2014], o problema $irr_f(G)$ é chamado no texto de *Target Set Selection*, e vemos que, além de generalizar o problema k-conjunto de conversão irreversível, $irr_f(G)$ generaliza outros problemas em grafos bem conhecidos, tais como *Dominating Set with thresholds*, *Vector Dominating Set*, *k-Tuple Dominating Set*, *Vertex Cover* (onde a função limiar descreve o valor do grau de cada vértice), *r-Neighbor Bootstrap Percolation* e *dynamic monopolies*.

A versão do problema com a função limiar $f(v, t)$ que muda com o tempo e que adiciona um tempo limite máximo para a infecção total do grafo é vista em Rautenbach et al. [2014]. Nesse artigo, vemos que se G é uma floresta ou um clique, Rautenbach et al. apresentam algoritmos eficientes que computem $irr(G, t_d, f)$. Além disso, são provados limites inferiores e superiores baseados em argumentos de contagem e probabilísticos. Para as escolhas especiais de t_d e f, o parâmetro $irr(G, t_d, f)$ coincide com parâmetros do grafo bem conhecidos relacionados à dominância e independência do grafo.

Porém, quando procuramos trabalhos na área experimental de processos de conversão irreversível, percebemos que estes são muito escassos na literatura. Em um dos poucos artigos experimentais encontrados, Amaral et al. utiliza algoritmos genéticos para encontrar resultados para o problema de conjunto convergente mínimo aqui visto. Utilizaremos algumas das instâncias criadas nesse artigo, e compararemos os resultados alcançados em Amaral et al. [2015] com nossas soluções para as instâncias propostas.

Heurísticas construtivas têm potencial de se adequarem muito bem ao problema proposto, tanto aquelas que levam em conta características locais de cada vértice, aqui chamada de heurística gulosa, quanto heurísticas mais elaboradas, que fazem a escolha de vértices baseadas no estado do grafo antes e após a inserção de um vértice ao conjunto infectado, que chamaremos de heurística de vértice efetivo. Estas heurísticas aqui propostas terão como *baseline* os resultados obtidos a partir da resolução de um modelo de programação inteira aqui proposto. O objetivo deste trabalho é propor uma alternativa para aproximar resultados deste problema conhecido na literatura por sua dificuldade, analisando a eficiência das abordagens aqui propostas para instâncias variadas, garantindo a robustez e corretude destas.

O restante deste trabalho está organizado da seguinte forma: A segunda seção apresenta a fundamentação teórica, onde processos de conversão irreversível são definidos formalmente, assim como o problema de conjunto convergente mínimo. Na seção 3, um modelo de programação inteira

para o problema é apresentado, e, em seguida, as heurísticas gulosa e vértice efetivo. A seção de metodologia experimental resume como foram construídas as instâncias e como os testes foram executados, bem como que ambiente de execução foi utilizado. Os resultados derivados do trabalho estão na seção 6, e em seguida, as conclusões finais.

2. Fundamentação teórica

Dado um grafo $G = (V, E)$, Centeno et al. analisam o problema PCI como uma sequência de rótulos binários

$$c_t : V(G) \to \{0, 1\} \tag{1}$$

$$C = (c_0, c_1 \dots) = (c_{t \in \mathbb{N}}) \tag{2}$$

e o modo como a característica se espalha pelo grafo é dada por uma função limiar f definida como:

$$f : V(G) \to \mathbb{Z}_+. \tag{3}$$

Dado um grafo G, uma função limiar f e $\mathcal{N}_G(u)$ conjunto de vizinhos de um dado vértice u, o processo começa com uma rotulação binária inicial c_0 dos vértices, enquanto o restante dos rótulos são definidos iterativamente de forma que $\forall t \in \mathbb{N}^*, \forall u \in V(G)$

$$c_t(u) = 1 \iff c_{t-1}(u) = 1 \vee \sum_{v \in \mathcal{N}_G(u)} c_{t-1}(v) \geq f(u). \tag{4}$$

Resumidamente, seja o conjunto inicial de vértices infectados c_0, um vértice não infectado será infectado se o total dos vizinhos contaminados for maior ou igual que o valor definido por f para este vértice. Um conjunto convergente é uma rotulação c_0 que determina uma certa sequência de rotulações c_t onde

$$\exists\, t_0 \geq 0 \mid \forall u \in V(G) : c_{t_0}(u) = 1, \tag{5}$$

e este conjunto é denotado por

$$c_0^{-1}(1) = \{u \in V(G) : c_0(u) = 1\} \tag{6}$$

Segue da definição que um conjunto convergente trivial é o próprio $V(G)$. O problema do Conjunto Convergente Mínimo (CCM) consiste em encontrar um conjunto convergente tão pequeno quanto possível.

Dada uma função limiar $f : V(G) \to \{k\}, k \in \mathbb{Z}_+$, Centeno et al. [2011] mostrou que este problema é NP-completo para qualquer $k \geq 2$. Como mesmo esta família de instâncias é de difícil resolução, este trabalho terá foco em problemas com função limiar constante. Ainda assim, instâncias onde o k varia de vértice para vértice também serão abordadas.

3. Modelagem do problema

Para uma modelagem por programação inteira, é interessante simplificar a notação e restrições dadas ao problema. Pensando nisso, a seguinte definição é uma maneira de representarmos o problema de maneira clara e objetiva: Seja $x_t(v)$ o rótulo de um vértice v no instante t. Para uma solução viável representando o problema CCM, é necessário garantir que

$$\begin{gathered} x_t(v) = 1 \iff x_{t-1}(v) = 1 \vee \sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) \geq f(v) \\ \forall t \in 1, \dots, |V| - 1, v \in V \end{gathered} \tag{7}$$

Note que se nenhum vértice puder ser infectado em um instante t, o processo de infecção estagnará, logo, um processo necessita de no máximo $|V|$ iterações para convergir. É por este motivo que na equação 7 a variável t está limitada a $|V|$ iterações, com exceção da inicial, que já está predefinida.

Consequentemente, queremos minimizar o número de rótulos infectados no instante 0:

$$min \sum_{v \in V} x_0(v) \tag{8}$$

sujeito às restrições a seguir, cada uma representando parte da definição formal descrita anteriormente. Para representar a restrição que um rótulo estar infectado no instante t, implica ele estar no instante $t+1$, ou seja

$$x_{t-1}(v) \implies x_t(v) \tag{9}$$

utilizaremos a desigualdade

$$x_{t-1}(v) \leq x_t(v). \tag{10}$$

Para indicar que um vértice com mais que $f(v)$ vértices infectados também será infectado, ou seja,

$$\sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) \geq f(v) \implies x_t(v) \tag{11}$$

o descreveremos como

$$\sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) - f(v) + 1 \leq |V| x_t(v) \tag{12}$$

Agora, para mostrar a implicação de volta, ou seja

$$x_t(v) = 1 \implies x_{t-1}(v) = 1 \vee \sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) \geq f(v) \tag{13}$$

ou sua contrapositiva

$$x_{t-1}(v) == 0 \wedge \sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) < f(v) \implies x_t(v) = 0 \tag{14}$$

será aplicada a seguinte restrição

$$\sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) + f(v) x_{t-1}(v) \geq f(v) x_t(v) \tag{15}$$

Finalmente, para que todos os vértices estejam infectados ao final do processo:

$$\sum_{v \in V} x_{|V|-1}(v) = |V| \tag{16}$$

Repare que o processo, neste caso, demorará no máximo $|V|$ iterações para convergir ao seu estado final. Assim, a modelagem completa do problema é dada a seguir.

$$\text{minimizar} \sum_{v \in V} x_0(v)$$

sujeito a

$$x_{t-1}(v) \leq x_t(v), \qquad t = \{1..|V|-1\}, v \in V$$

$$\sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) - f(v) + 1 \leq |V| x_t(v), \qquad t = \{1..|V|-1\}, v \in V$$

$$\sum_{u \in \mathcal{N}_G(v)} x_{t-1}(u) + f(v) x_{t-1}(v) \geq f(v) x_t(v), t = \{1..|V|-1\}, v \in V$$

$$\sum_{v \in V} x_{|V|-1}(v) = |V|$$

$$x_t(v) \in \{0,1\}, \qquad t = \{1..|V|-1\}, v \in V$$

4. Heurísticas

Heurísticas construtivas são uma abordagem muito interessante para o problema de conjunto convergente mínimo, pois soluções incrementais, neste caso, são simples e a maneira mais intuitiva de atacá-lo. A seguir, são descritas as duas abordagens criadas neste artigo.

4.1. Heurística Gulosa (HG)

A ideia por traz desta heurística é escolher localmente os vértices a serem infectados. Os fatores que podem ser levados em conta na hora da decisão são:

(a) grau do vértice (HG): Vértices de maior grau tendem a infectar um maior número de vértices;

(b) valor da função limiar para o vértice (HGL): Um vértice tem menos chances de ser infectado ”naturalmente”(por uma vizinhança infectada) se o valor de sua função limiar for muito grande. Pensando nisso, uma segunda ideia seria usar ambas as estratégias, e ordenar os vértices de acordo com seu grau e função limiar, seguindo a seguinte fórmula: $\mathcal{N}_G(v) * f(v)$. Obviamente, os casos com estratégia relacionada a função limiar só serão testados em instâncias com função limiar não constante.

4.2. Heurística Vértice Efetivo (HVE)

O objetivo do problema CCM é encontrar o menor número de vértices que infectem todo o grafo ao longo de um processo. Assim, temos a ideia que, a cada intervalo de tempo t, seja escolhido um vértice que maximiza o número de novos vértices infectados, e que, caso ocorra empate, uma estratégia que não considere somente este número de infecções seja aplicada.

Assim, a cada vértice não infectado, mas com vizinhos infectados, foi atribuído um contador de vizinhos doentes, tal que, quando este atingir o valor da função limiar, o vértice é infectado e o contador não mais incrementado. Pensando nisso, a heurística escolherá o vértice considerando aquele que mais causou infecções, e em caso de empate, escolherá o vértice que mais aumentou a soma dos valores dos contadores acima citados. Ou seja, tenta-se escolher o vértice mais efetivo no grafo a cada intervalo de tempo t, que mais fará progresso nesse instante.

5. Metodologia experimental

A área de Processos de Conversão Irreversível possui muitos artigos com resultados teóricos, como Dreyer e Roberts [2009]; Centeno et al. [2011], porém pouca pesquisa experimental. O único trabalho experimental na área foi feito por Amaral et al. [2015]. Utilizaremos algumas das ideias por trás de um conjunto de instâncias criadas nesse artigo na etapa de experimentação do trabalho. Serão também utilizadas instâncias de problemas de coloração fornecidas pelo projeto DIMACS Challenge [Trick] e instâncias geradas randomicamente.

Antes de qualquer heurística ser aplicada neste trabalho, iremos testar a existência de vértices em que seu grau de vizinhança é menor que sua resistência a infecções, ou seja, vértices que não possuem vizinhos suficientes para serem infectados, e por isso, obrigatoriamente estarão em qualquer conjunto convergente mínimo inicialmente infectado. Esta operação nos custa tempo apenas linear no tamanho do grafo, visto que o grau de cada vértice é pré-computado na leitura inicial do arquivo.

5.1. Instâncias para teste

As instâncias utilizadas neste trabalho são categorizadas em três grupos distintos:

1. Instâncias de Coloração: Serão criadas instâncias a partir das instâncias DIMACS de problemas de coloração, que serão utilizadas principalmente para mostrar o desempenho das heurísticas quando comparadas com os resultados obtidos com a resolução do modelo de programação inteira pelo resolvedor. Estas instâncias são criadas partindo do grafo do problema de coloração, e utilizando duas estratégias para a criação da função limiar:

   (a) Uma será constante com $f(v) = 2, \forall v \in V$

   (b) A outra será proporcional ao grau de cada vértice, com $f(v) = \lfloor \frac{\text{grau}(v)}{2} \rfloor, \forall v \in V$.

   O número de vértices em cada uma destas instâncias variam entre 10 e 1.000, inclusive.

2. Grafos randômicos: Serão utilizados nesta comparação, mostrando como os resultados são afetados pela diferença entre construções aleatórias e grafos mais bem estruturados. Sua geração utiliza um parâmetro para a densidade de arestas desejada, ou seja, quantas das $O(n^2)$ possíveis arestas serão criadas, aleatoriamente, na instância. A estratégia para criar a função limiar é a mesma usada anteriormente para os grafos de coloração:

   (a) Uma será constante com $f(v) = 2, \forall v \in V$

   (b) A outra será proporcional ao grau de cada vértice, com $f(v) = \lfloor \frac{\text{grau}(v)}{2} \rfloor, \forall v \in V$.

   Os nomes dados às estas instâncias estarão no formato:
   <ordem do grafo>-<densidade do grafo em porcentagem>-<função limiar escolhida>rand

3. Instâncias projetadas: Criadas por Amaral et al. [2015], nada mais são que grafos arbitrariamente grandes, projetados para ter um conjunto convergente mínimo de tamanho 2, dado qualquer número de vértices. Este tipo de instância será interessante para investigarmos a eficiência das heurísticas para tamanhos de grafo mais elevados, no qual um resolvedor não seria uma opção válida para comparação de resultados. A ideia por trás desta construção projetada é replicar diversas vezes o mesmo grafo e unir arestas de maneira encadeada, como na Figura 5.1.

Amaral et al. analisa também instâncias em árvores binárias, que não serão interessantes para as heurísticas aqui propostas. Visto que um dos primeiros passos nos experimentos, antes mesmo de executar as heurísticas, é infectar quaisquer vértices do grafo que obrigatoriamente farão parte da solução, por não conseguirem ser infectados de outra forma, os casos de árvores binárias propostos seriam resolvidos otimalmente sem mesmo passar pela heurística, o que não é nosso objetivo neste trabalho. Além disso, o algoritmo desenvolvido em Centeno et al. [2011] resolve o problema CCM para esta instância em árvores com $k \leq 2$ em tempo polinomial.

5.2. Ambiente de Execução

A implementação das heurísticas construtivas foi feita em C++11, enquanto o resolvedor de programação linear inteira CPLEX(R) Interactive Optimizer 12.7.0.0 foi utilizado pra resolver os modelos. A modelagem para o resolvedor CPLEX foi feita utilizando a API para Python suportada pela plataforma. Os testes foram realizados em um computador Dell Inspiron 15 5000 series com processador Intel Core i7-4510U CPU, 16 GB de RAM e Sistema Operacional Ubuntu 16.04.2 LTS, kernel Linux 4.4.0-79-generic x86_64 .

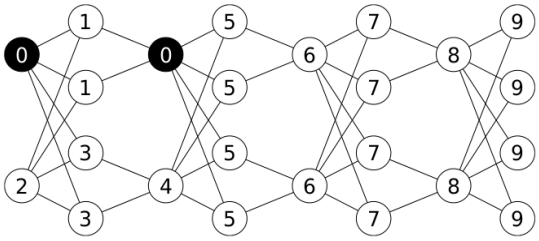

Figura 1: Exemplo de instância projetada com 4 camadas, cada uma com 6 vértices. Fonte: Amaral et al. [2015].

6. Resultados

Nesta seção, mostraremos os resultados obtidos para as instâncias propostas, dando atenção ao tempo de execução e qualidade das soluções. Primeiramente serão comparados os desempenhos das duas heurísticas com o do resolvedor para problemas de até 1.000 vértices, para as quais o resolvedor ainda consegue resolver ou aproximar o problema relativamente bem. Em seguida, apresentaremos o desempenho das heurísticas para grafos maiores, da ordem de até 14.500 vértices, porém, sem os resultados do resolvedor, visto que o tempo de execução do mesmo fica inviável, mesmo para aproximações.

6.1. Comparando heurísticas e resolução exata do modelo

Primeiramente serão usadas as 56 instâncias de coloração modificadas, grupo (1), usando dois tipos de função limiar: Primeiramente será usado o grupo (1.a), com $f(v) = 2$, e, após, o grupo (1.b), onde $f(v) = deg(v)/2$. Dada a melhor solução s_0 entre as encontradas, calcula-se o quão distante as outras soluções s estão desta, i.e, $(100 * (s_0 - s)/s_0)\%$ que será medido em porcentagem e chamaremos aqui de distância da melhor solução encontrada, ou abreviadamente DMSE. São calculados também o tempo, em segundos, e o tamanho das soluções encontradas, que neste caso é o número de vértices infectados inicialmente que infectam todo o grafo. Deixando claro, o tempo de execução do resolvedor foi limitado a um máximo de 70 minutos por instância, e se a solução ótima não é encontrada, a melhor solução viável encontrada é considerada como referência para o cálculos dos DMSEs e o tempo de execução na tabela é deixado em branco. As unidades de medida utilizadas na tabela são (v) para número de vértices, (s) para segundos e (%) para porcentagem, e as duas tabelas disponíveis possuem apenas parte dos resultados, porém as médias são de todos os dados.

Na Tabela 1, observamos que os resultados de HVE são os melhores encontrados para as instâncias de coloração modificadas aqui propostas, tendo, em todos os casos, o menor número de vértices que resolvem o problema CCM. O resolvedor tem resultados piores quando comparado com os resultados de HVE, mas ainda assim, obteve solução ótima para algumas das instâncias. No entanto, seu DMSE médio foi o mais alto, pois grande parte das instâncias não puderam ser resolvida em 70 minutos, e nesses casos, as aproximações não foram boas. HG obteve muitos resultados que coincidem com o melhor resultado encontrado, porém para alguns casos, obteve DMSE muito grande, fazendo com que estes *outliers*, no caso, 6 instâncias de todas as listadas, aumentem o DMSE e o número médio de vértices na solução em muito. Além disso, quando olhando o tempo de execução de HG e HVE para estas instâncias, vemos que estes são reduzidos, da ordem de milesegundos em média, principalmente quando comparado com o resolvedor, com média de 10 minutos por solução encontrada, fora as que ultrapassaram os 70 minutos.

Em comparação, os resultados obtidos para as instâncias com função limiar proporcional ao grau do vértice podem ser vistos na Tabela 2. Nota-se que o problema fica consideravelmente mais balanceado, porém mais custoso computacionalmente, para esta instanciação com função limiar não constante. A heurística de vértice efetivo ainda superou as outras abordagens, porém,

Instâncias	Solução CPLEX	Tempo CPLEX	CPLEX DMSE	Solução HG	Tempo HG	HG DMSE	Solução HVE	Tempo HVE	HVE DMSE
anna	28	1,135	0,00%	125	0,01	346,43%	28	0,002	0,00%
fpsol2.i.1	257	-	12,23%	229	0,001	0,00%	229	0,059	0,00%
fpsol2.i.3	64	944,382	0,00%	64	0,001	0,00%	64	0,061	0,00%
huck	11	5,846	0,00%	70	0,003	536,36%	11	0,001	0,00%
inithx.i.2	150	-	68,54%	89	0,001	0,00%	89	0,157	0,00%
jean	21	5,868	0,00%	76	0,003	261,90%	21	0,001	0,00%
le450_15b	51	-	1600,00%	4	0,001	33,33%	3	0,003	0,00%
le450_15c	2	356,301	0,00%	2	0,002	0,00%	2	0,074	0,00%
le450_25a	2	381,316	0,00%	2	0,001	0,00%	2	0,072	0,00%
le450_25c	26	-	1200,00%	2	0,002	0,00%	2	0,09	0,00%
le450_5b	47	-	2250,00%	2	0,001	0,00%	2	0,073	0,00%
le450_5d	2	514,701	0,00%	2	0,001	0,00%	2	0,077	0,00%
miles1500	2	-	100,00%	2	0,001	0,00%	2	0,01	0,00%
miles500	9	-	350,00%	83	0,012	0,00%	2	0,007	0,00%
mulsol.i.1	61	1060,294	0,00%	61	0,001	0,00%	61	0,013	0,00%
mulsol.i.3	12	1098,787	0,00%	12	0	0,00%	12	0,015	0,00%
mulsol.i.5	12	1108,467	0,00%	12	0,001	0,00%	12	0,015	0,00%
myciel4	2	6,826	0,00%	2	0	0,00%	2	0	0,00%
myciel6	2	1150,961	0,00%	2	0	0,00%	2	0,006	0,00%
queen10_10	2	649,191	0,00%	2	0	0,00%	2	0,007	0,00%
queen12_12	2	1084,953	0,00%	2	0	0,00%	2	0,008	0,00%
queen14_14	2	1050,155	0,00%	2	0,001	0,00%	2	0,017	0,00%
queen16_16	2	986,701	0,00%	2	0,001	0,00%	2	0,026	0,00%
Média dos resultados	**35,96 vértices**	**656,7 min**	**326%**	**46,79 vértices**	**0,01 s**	**110%**	**24,34 vértices**	**0,06 s**	**0%**

Tabela 1: Resultados obtidos para parte das instâncias (1.a) com as heurísticas gulosa (HG) e vértice efetivo (HVE). Comparadas com os resultados obtidos pelo resolvedor CPLEX. Médias de todas as 56 instâncias.

diferente do caso anterior, não há tanta diferença em média entre os resultado. O tempo consumido pelo algoritmo guloso, no entanto, é significativamente melhor que os outros, sendo em média ainda da ordem de milisegundos, enquanto HVE agora está em segundos, e o resolvedor atinge o limite de tempo para a maioria das instâncias, sem conseguir alcançar o resultado ótimo.

Agora, para grafos ainda bem estruturados, mas com resultado ótimo baixo, foram resolvidas as instâncias (3), que são projetadas para ter, como visto anteriormente, solução constante igual a 2. Na Tabela 3 vemos os resultados obtidos a partir destas instâncias e podemos observar que a heurística gulosa não atingiu resultados satisfatórios. É interessante notar que tamanho dos resultados é muito bem comportado, sendo sempre de tamanho $N/3 - 1$, o que é esperado, tendo em vista maneira como o grafo é construído: Quando aumentamos o número de vértices, apenas replicamos a grafo de novo, logo os vértices de maior grau continuam os mesmos; além disso, o algoritmo escolhe os vértices em uma ordem particularmente ruim, escolhendo o vértice que alastra melhor a infecção apenas após uma quantidade significativa de vértices. Apesar de, em média, ser duas vezes mais lento que a heurística gulosa, HVE teve desempenho muito bom, pois os resultados obtidos foram exatamente a solução ótima do problema CCM para as dadas instâncias. Isto indica, juntamente com resultados anteriores, que esta heurística é a que melhor se adequa a instâncias com grafos bem estruturados.

Assim, é interessante compararmos estes resultados com os obtidos em Amaral et al. [2015] com o uso de algoritmos genéticos. Amaral et al. registram o tempo de execução médio que seu algoritmo leva para encontrar a solução ótima da instância projetada, e na Tabela 4 vemos, além disso, o tempo que HVE, em comparação, demora para resolver otimamente as mesmas instâncias. Nota-se que nossa heurística é seis ordens de grandeza mais rápida para o maior caso apresentado, e, para o menor, 4 ordens, o que é uma boa melhora na solução do problema.

Resultados obtidos para instâncias construídas com grafos randômicos podem fornecer

Instâncias	Solução CPLEX	Tempo CPLEX	CPLEX DMSE	Solução HG	Tempo HG	HG DMSE	Solução HVE	Tempo HVE	HVE DMSE
anna	6	-	200,00%	3	0	50,00%	2	0,012	0,00%
fpsol2.i.1	368	-	35,79%	327	0,052	20,66%	271	1,138	0,00%
fpsol2.i.3	261	-	210,71%	102	0,016	21,43%	84	0,989	0,00%
huck	10	4106,519	0,00%	70	0,004	600,00%	10	0,013	0,00%
inithx.i.2	377	-	245,87%	119	0,022	9,17%	109	2,006	0,00%
jean	12	-	9,09%	41	0,002	272,73%	11	0,011	0,00%
le450_15b	206	-	171,05%	76	0,019	0,00%	123	6,238	61,84%
le450_15d	232	-	90,16%	122	0,043	0,00%	165	8,516	35,25%
le450_25b	204	-	251,72%	58	0,011	0,00%	105	5,046	81,03%
le450_25d	226	-	107,34%	109	0,04	0,00%	151	7,717	38,53%
le450_5b	227	-	118,27%	104	0,022	0,00%	128	6,11	23,08%
le450_5d	233	-	87,90%	124	0,025	0,00%	156	7,347	25,81%
miles1500	65	-	75,68%	37	0,01	0,00%	50	0,248	35,14%
miles500	52	-	100,00%	112	0,023	330,77%	26	0,121	0,00%
mulsol.i.1	129	-	37,23%	94	0,004	0,00%	94	0,204	0,00%
mulsol.i.3	98	-	206,25%	39	0,004	21,88%	32	0,156	0,00%
mulsol.i.5	103	-	221,88%	40	0,004	25,00%	32	0,163	0,00%
myciel4	3	104,264	0,00%	3	0	0,00%	3	0	0,00%
myciel6	33	-	175,00%	12	0,001	0,00%	12	0,023	0,00%
queen10_10	40	-	11,11%	39	0,004	8,33%	36	0,096	0,00%
queen12_12	71	-	26,79%	58	0,011	3,57%	56	0,267	0,00%
queen14_14	99	-	26,92%	79	0,024	1,28%	78	0,839	0,00%
queen16_16	152	-	43,40%	110	0,044	3,77%	106	1,845	0,00%
Média dos resultados	**132,446 vértices**	**1712,7 min**	**96,88%**	**89,035 vértices**	**0,023 s**	**56,48%**	**77 vértices**	**1,954 s**	**11,35%**

Tabela 2: Resultados obtidos para parte instâncias (1.b) com as heurísticas gulosa (HG) e vértice efetivo (HVE). Comparadas com os resultados obtidos pelo resolvedor CPLEX. Médias de todas as 56 instâncias.

Ordem do grafo (v)	Solução HG (v)	Tempo HG (s)	Solução HVE (v)	Tempo HVE (s)
600	199	0,04	2	0,08
1200	399	0,22	2	0,42
1800	599	0,48	2	0,95
2400	799	0,86	2	1,69
3000	999	1,36	2	2,70
3600	1199	1,96	2	3,86
4200	1399	2,64	2	5,31
4800	1599	3,49	2	6,90
5400	1799	4,55	2	8,85
6000	1999	5,00	2	10,41
6600	2199	6,82	2	13,35
7200	2399	8,40	2	15,99
7800	2599	9,61	2	18,39
8400	2799	11,01	2	21,39
9000	2999	12,70	2	24,77

Tabela 3: Resultados obtidos para instâncias (3) com as heurísticas gulosa (HG) e vértice efetivo (HVE).

mais informações em relação à qualidade e desempenho dos algoritmos aqui propostos, principalmente o desempenho em grafos menos estruturados. Na Tabela 5, vemos as instâncias (2) e os resultados obtidos, onde observamos que, desta vez, HG obteve melhores resultados que HVE, tanto em tempo, quanto nas soluções, com exceção de uma instância, mostrando que a heurística gulosa se comporta muito bem para estes grafos. Infelizmente, a estratégia gulosa limiar obteve os mesmos resultados, ou piores, que a estratégia gulosa padrão, não acrescentando em nada aos resultados obtidos. Os resultados com o resolvedor ainda são os piores, tendo a maior média de número de vértices por solução e maior DMSE médio, além de ter passado do limite de tempo para

Ordem do grafo (v)	Tempo médio de convergencia (s) Amaral et al. [2015]	Tempo HVE (s)
60	19,6	0,0011
126	71,3	0,0056
252	253,6	0,0223
510	1440,7	0,0818
1020	8005,6	0,3235
2046	43382,5	1,2618
4092	246260,0	5,2374

Tabela 4: Comparando resultados para instâncias (3) com as heurísticas vértice efetivo (HVE) e os apresentados em Amaral et al. [2015].

todas as instâncias.

Nesta seção, obtivemos resultados com o resolvedor para instâncias menores que 1.000 vértices, pois para instâncias maiores, este demora muitas horas apenas no pré-processamento dos dados da instância, e demoraria ainda mais no processamento em si. Por isso, na seção a seguir, abandonaremos tentativas de resultados com o resolvedor, e focaremos nas heurísticas aqui propostas aplicadas a instâncias maiores.

Instâncias	Sol CPLEX (v)	Tempo CPLEX (s)	CPLEX DMSE	Sol HG (v)	Tempo HG (s)	HG DMSE	Sol HGL (v)	Tempo HGL (s)	HGL DMSE	Sol HVE (v)	Tempo HVE (s)	HVE DMSE
200-35-2rand	61	-	110,34%	29	0,003	0,00%	29	0,003	0,00%	30	0,293	3,45%
200-35-4rand	30	-	233,33%	9	0,001	0,00%	9	0,001	0,00%	10	0,101	11,11%
400-60-2rand	60	-	62,16%	37	0,004	0,00%	37	0,005	0,00%	37	0,394	0,00%
400-60-4rand	31	-	93,75%	16	0,002	0,00%	16	0,002	0,00%	16	0,180	0,00%
600-85-2rand	56	-	30,23%	43	0,007	0,00%	43	0,007	0,00%	44	0,482	2,33%
600-85-4rand	29	-	38,10%	21	0,004	0,00%	21	0,003	0,00%	21	0,239	0,00%
800-35-4rand	35	-	288,89%	25	0,008	177,78%	10	0,004	11,11%	9	0,575	0,00%
800-60-2rand	63	-	530,00%	10	0,003	0,00%	35	0,014	250,00%	36	2,896	260,00%
800-60-4rand	34	-	112,50%	16	0,009	0,00%	16	0,007	0,00%	16	1,057	0,00%
Médias	**44,3**	**-**	**166,59%**	**11,4**	**0,005**	**19,75%**	**24**	**0,005**	**29,01%**	**24,3**	**0,691**	**30,76%**

Tabela 5: Resultados obtidos para instâncias (2) com as heurísticas gulosa (HG), gulosa limiar(HGL) e vértice efetivo (HVE).

6.2. Resultados das Heurísticas para Grafos Maiores

Utilizando grafos randômicos de tamanhos variando entre 1.000 e 14.500, nesta seção focaremos somente na comparação entre as heurísticas aqui propostas. Na Tabela 6, vemos os resultados obtidos para instâncias com função limiar constante igual a 2. Apesar deste ser um problema NP-difícil, nota-se que os resultados obtidos são todos ótimos, visto que um limite inferior para o problema CCM com função limiar constante k, no caso $k = 2$, é a própria constante k. O tempo de execução para HVE é bem maior que o das outras heurísticas, demorando por volta de 45 minutos para a maior instância enquanto HG demora segundos, o que mostra a vantagem que HG tem para grafos randômicos até o momento. Note também que o tempo de execução de HGL foi ligeiramente melhor que o de HG, com a mesma qualidade de solução, mas ainda assim, esta versão da heurística não gera resultados tão interessantes, ainda mais quando comparada à HG e HVE.

Instâncias randômicas com função limiar proporcional ao grau dos vértices são muito mais difíceis que a anteriormente vista, e por isso, o custo computacional para calcular os resultados aumenta em muito. Assim, teremos duas tabelas, a primeira com HVE e HG, visto que HGL não acrescenta em muito na análise, e a segunda somente com HG, pois HVE fica muito mais caro computacionalmente para instâncias maiores que 4.000 vértices, levando mais de 4 horas por instância. Assim, temos a Tabela 7, que mostra os resultados alcançáveis em tempo hábil para

Instância	Resultado HG (v)	Tempo HG (s)	Resultado HGL (v)	Tempo HGL (s)	Resultado HVE (v)	Tempo HVE (s)
1000-25-1rand	2	0,006	2	0,005	2	0,514
1000-40-1rand	2	0,008	2	0,008	2	0,769
4000-10-1rand	2	0,033	2	0,034	2	17,027
4000-25-1rand	2	0,092	2	0,092	2	35,191
4000-40-1rand	2	0,162	2	0,159	2	60,457
7000-10-1rand	2	0,117	2	0,118	2	77,295
7000-40-1rand	2	0,344	2	0,426	2	267,375
10000-25-1rand	2	0,495	2	0,552	2	552,871
10000-40-1rand	2	0,746	2	0,948	2	802,096
13000-10-1rand	2	0,392	2	0,409	2	522,529
13000-40-1rand	2	1,154	2	1,286	2	1585,060
14500-10-1rand	2	0,480	2	0,477	2	587,420
14500-25-1rand	2	0,927	2	0,900	2	1576,810
14500-40-1rand	2	2,018	2	1,712	2	2742,310

Tabela 6: Resultados obtidos para instâncias (2.a) onde $f(v) = 2$ com as heurísticas gulosa (HG) e vértice efetivo (HVE).

instâncias até este tamanho. Observe que, apesar de um DMSE baixo, os resultados e tempo de execução de HVE forem todos piores que HG. Ainda mais, é interessante ver na Tabela 8 que HG consegue resolver instâncias de até 14.500 vértices em tempo relativamente baixo, com menos de 35 minutos para a maior das instância testada. Vemos, assim, que HG parece ser a melhor escolha para a resolução do grupo de instâncias (2), que representam aqui instâncias menos estruturadas.

Instâncias	Resultados HG (v)	Tempo HG (s)	HG DMSE	Resultado HVE (v)	Tempo HVE (s)	HVE DMSE
01000-10-2rand	352	0,349	0,00%	379	132,428	7,67%
01000-25-2rand	410	1,055	0,00%	426	223,039	3,90%
02500-40-2rand	435	1,389	0,00%	440	319,841	1,15%
04000-10-2rand	1010	5,641	0,00%	1059	4019,890	4,85%
04000-25-2rand	1107	14,768	0,00%	1133	8726,550	2,35%

Tabela 7: Resultados obtidos para instâncias (2.b) até 4.000 vértices onde $f(v) = \lfloor \frac{\text{grau}(v)}{2} \rfloor$ com as heurísticas gulosa (HG), gulosa limiar (HGL) e vértice efetivo (HVE).

Instâncias	Resultados HG (v)	Tempo HG (s)
04000-40-2rand	1861	95,309
07000-10-2rand	3057	121,259
07000-25-2rand	3244	318,318
07000-40-2rand	3301	508,585
10000-10-2rand	4476	354,208
10000-25-2rand	4684	909,631
10000-40-2rand	4771	1518,360
13000-10-2rand	5168	539,686
13000-25-2rand	5413	1421,150
13000-40-2rand	5499	2305,120
14500-10-2rand	5899	792,752
14500-25-2rand	6144	2047,380

Tabela 8: Resultados obtidos para instâncias (2.b) com mais de 4.000 vértices onde $f(v) = \lfloor \frac{\text{grau}(v)}{2} \rfloor$ com a heurística gulosa (HG).

7. Conclusões Finais

A partir das hipóteses testadas neste trabalho, concluímos que ambas heurísticas propostas foram muito bem sucedidas em encontrar soluções de qualidade para as instâncias aqui vistas.

Também, vimos que ambas foram complementares em seus resultados, cada uma obtendo soluções melhores para uma determinada classe de instâncias.

A heurística vértice efetivo teve os menores resultados consistentemente para todas as instâncias de coloração, bem como para as instâncias projetadas, na qual obteve resultados rápida e efetivamente. Por outro lado, seu custo computacional é mais elevado, principalmente para grafos mais gerais, enquanto comparado com a heurística gulosa. Enquanto isso, a heurística gulosa se mostrou extremamente rápida, demorando poucos segundos mesmo para as instâncias maiores, da ordem de 14.500 vértices. Obteve os menores resultados para grafos randômicos, tanto com função limiar constante, como não. No entanto, a variação de HG, que denominamos HGL e que considera o valor da função limiar na escolha local, não mostrou-se vantajosa para a maioria das instâncias testadas.

Com relação ao modelo de programação inteira para o problema que utilizamos como *baseline*, este foi extremamente lento, o que é esperado de uma primeira modelagem direta de um problema conhecidamente NP-difícil. Porém, há ainda diversos caminhos a serem explorados: A busca por novas restrições que reduzam o espaço de solução, a proposição de novos limites duais, ou até uma nova abordagem e maneira de modelar o problema poderiam acelerar em muito sua resolução. Além disso, usar como solução inicial os resultados obtidos com as heurísticas construtivas aqui propostas, e a partir dai melhorá-las, é uma boa maneira de resolver problemas de forma exata e de maneira mais rápida.

Futuramente, seria interessante explorar diferentes tipos de instâncias. Por exemplo, estudar o quão bem a heurística vértice efetivo se comporta com outros tipos de grafos bem estruturados, como grafos das classes cordal, de comparabilidade ou permutação, por exemplo. Explorar novas heurísticas e maneiras de atacar o problema seria também interessante. Enfim, acreditamos existir espaço para mais trabalhos práticos para este e outros problemas relacionados a Processos de Conversão Irreversível, tendo em vista o extensivo trabalho teórica disponível, comparado com a quase inexistente bibliografia experimental na área.

Referências

Amaral, A. A. T., Wanner, E. F., e dos Santos, V. F. (2015). Resolução de problemas de conversão irreversível por meio de algoritmos genéticos. In Bastos Filho, C. J. A., Pozo, A. R., e Lopes, H. S., editors, *Anais do 12 Congresso Brasileiro de Inteligência Computacional*, p. 1–6, Curitiba, PR. ABRICOM.

Centeno, C. C., Dourado, M. C., Penso, L. D., Rautenbach, D., e Szwarcfiter, J. L. (2011). Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693–3700.

Chopin, M., Nichterlein, A., Niedermeier, R., e Weller, M. (2014). Constant thresholds can make target set selection tractable. *Theory of Computing Systems*, 55(1):61–83. ISSN 1433-0490. URL `http://dx.doi.org/10.1007/s00224-013-9499-3`.

Dreyer, P. A. e Roberts, F. S. (2009). Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627.

Rautenbach, D., dos Santos, V. F., e Schäfer, P. M. (2014). Irreversible conversion processes with deadlines. *Journal of Discrete Algorithms*, 26:69–76.

Reddy, T. V. T., Krishna, D. S., e Rangan, C. P. (2010). *Variants of Spreading Messages*, p. 240–251. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-11440-3. URL `http://dx.doi.org/10.1007/978-3-642-11440-3_22`.

Trick, M. DIMACS CHALLENGE : Graph coloring instances.