Procedimentos de Redução para o Problema da Máxima Interseção de k-Subconjuntos

Fábio C. S. Dias
Campus Quixadá – Universidade Federal do Ceará (UFC)
Av. José de Freitas Queiroz, 5003, Cedro. Quixadá - CE
fabiodias@ufc.br

José Robertty de Freitas Costa
Departamento de Computação – Universidade Federal do Ceará (UFC)
Av. Humberto Monte, s/n. Bloco 902, Campus do Pici. Fortaleza - CE
robertty@alu.ufc.br

Wladimir Araújo Tavares
Campus Quixadá – Universidade Federal do Ceará (UFC)
Av. José de Freitas Queiroz, 5003, Cedro. Quixadá - CE
wladimirtavares@ufc.br

RESUMO

O Problema da Máxima Interseção de k-Subconjunto é definido como: dado uma coleção $\mathcal{S}$ de subconjuntos do conjunto finito B e um inteiro k, deseja-se selecionar k subconjuntos $S_1, S_2, \ldots, S_k \in \mathcal{S}$, tal que $|S_1 \cap \ldots \cap S_k|$ seja máxima. Na literatura, existem procedimentos de redução que visam remover vértices do grafo de entrada que não estarão em uma solução ótima de custo maior ou igual a uma solução viável conhecida. Neste trabalho, apresentamos um novo procedimento de redução e uma estratégia de redução que pode ser combinada com os procedimentos já existentes na literatura. Realizamos exaustivos experimentos computacionais para encontrar o melhor procedimento de redução. Os resultados dos testes computacionais mostram que a estratégia de redução combinada com procedimento de redução menos efetivo supera o melhor procedimento de redução.

PALAVRAS CHAVE. Problema da Máxima Interseção de k-Subconjuntos, Procedimento de redução, Contração de vértice.

Tópicos: Otimização Combinatória.

ABSTRACT

The Maximum k-Subset Intersection problem is defined as: given a collection $\mathcal{S}$ of subsets of the finite set B and an integer k, we have to select k subsets $S_1, S_2, \ldots, S_k \in \mathcal{S}$, such that $|S_1 \cap \ldots \cap S_k|$ is maximum. In the literature, there are reduction procedures that aim to remove vertices from the input graph that will not be in an optimal solution with a cost greater than or equal to a known feasible solution. In this work, we present a new reduction procedure and a strategy that can be combined with existing procedures in the literature. We performed exhaustive computational experiments to find the best reduction procedure. The results of these evaluations show that the reduction strategy combined with the least effective reduction procedure outperforms the best reduction procedure.

KEYWORDS. k-Subset Maximum Intersection Problem, Reduction procedures.

Topics: Combinatorial Optimization.

1. Introdução

Muitos problemas de otimização podem ser definidos quando temos um conjunto de indivíduos determinados por um conjunto de características e, desejamos maximizar o maior número de indivíduos com o maior número de características em comum, ou ainda, encontrar o maior número de indivíduos que compartilham uma certa quantidade de características em comum. Esses problemas surgem naturalmente em diversas aplicações como: controle de divulgação de dados [Vinterbo, 2002], análise da interação entre genes e condições (como ambiente, drogas, doenças, etc) [Nussbaum et al., 2012], análise da preferência musical de usuários em redes sociais [Bogue et al., 2014] e agrupamento de produtos em pacotes de acordo com a demanda dos clientes [Acuña et al., 2014].

Um dos problemas mais estudados dessa família é o Problema da Biclique Máxima em Arestas (MEBP, do inglês *Maximum Edge Biclique Problem*), onde o objetivo é encontrar uma biclique em um grafo com o número máximo de arestas. Diversas variantes para este problema foram considerados na literatura, entre elas, o Problema da Máxima Interseção de k-Subconjunto (kMIS, do inglês *Maximum Intersection of k-Subsets*). Nesta variante, consideramos a restrição de selecionar apenas k vértices de uma das partições do grafo bipartido.

Formalmente, o Problema da Máxima Interseção de k-Subconjuntos (kMIS) pode ser definido como: Dado uma coleção $\mathcal{S}$ de subconjuntos do conjunto finito B e um inteiro k, deseja-se selecionar k subconjuntos $S_1, S_2, \ldots, S_k \in \mathcal{S}$, tal que $|S_1 \cap \ldots \cap S_k|$ seja máxima.

O problema pode ser modelado com um grafo bipartido $G = (L \cup R, E)$, sendo que cada vértice $u \in L$ está associado a um subconjunto $S_u \in \mathcal{S}$ e cada vértice $v \in R$ está associado a um elemento $b_v \in B$. Existe uma aresta entre um vértice u (associado a um subconjunto $S_u \in \mathcal{S}$) e um vértice v (relacionado a um elemento $b_v \in B$), se e somente se, $b_v \in S_u$. Portanto, queremos determinar $L' \subset L$, com $|L'| = k$, de forma que $|\cap(L')|$ seja máximo, onde $\cap(L') = \bigcap_{u \in L'} N_G(u)$, com $N_G(u)$ sendo a vizinhança de u em G. Por simplicidade, iremos cometer o seguinte abuso de notação: algumas vezes vamos nos referir aos vértices de L pelos subconjuntos $S_u \in \mathcal{S}$ e mesmo denomina-los vértices ou de subconjuntos.

Considere o seguinte exemplo, $B = \{1, 2, 3, 4, 5\}$ representando o conjunto de características dos indivíduos, $\mathcal{S} = \{S_1, S_2, S_3\}$ representando o conjunto de indivíduos, onde $S_1 = \{2, 3, 4\}$, $S_2 = \{1, 4, 5\}$ e $S_3 = \{1, 2, 3, 4\}$. Para $k = 2$, o problema consiste em selecionar dois indivíduos com o maior número de características em comum . A Figura 1 ilustra esse exemplo. Note que $|S_1 \cap S_2| = 1, |S_1 \cap S_3| = 3$ e $|S_2 \cap S_3| = 2$. Portanto, a solução ótima para essa instância é dada por $L' = \{S_1, S_3\}$. Observe que o subgrafo induzido pelos vértices S_1 e S_3 e todos os vértices associados com os elementos em $S_1 \cap S_3$ definem um grafo bipartido completo.

Diversos resultados teóricos foram apresentados para o kMIS. Em [Vinterbo, 2002] e [Xavier, 2012], o problema kMIS foi mostrado ser $\mathcal{NP}$-Difícil. [Xavier, 2012], provou que kMIS não pode ser aproximado dentro um fator constante $\frac{1}{n^\varepsilon}$, onde n é o número de subconjuntos e $\varepsilon > 0$, considerando que $NP \not\subset BPTIME(2^{n^\varepsilon})$. [Shieh et al., 2012], mostrou que kMIS não pode ser aproximado dentro um erro absoluto $\frac{1}{2}n^{1-2\varepsilon} + O(n^{1-3\varepsilon})$ a menos que $P = NP$.

O problema kMIS tem sido bastante estudado de diferentes perspectivas, usando abordagens exatas e heurísticas. Em particular, temos um interesse especial nos trabalhos que apresentam procedimento de redução do problema. Em [Bogue et al., 2014], os autores apresentam um procedimento de pré-processamento visando reduzir o tamanho da entrada do problema através da eliminação de vértices de L e R que não podem fazer parte de uma solução viável com o custo maior que um limite inferior λ. Este processo pode ser reiterado até que nenhum vértice possa removido do grafo associado ao problema. Em algumas instâncias reais, os autores relatam que

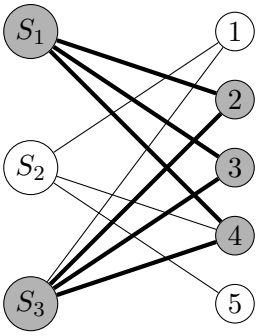


Figura 1: Para $k = 2$, a máxima interseção de k subconjuntos será obtido com os subconjuntos S_1 e S_3. O tamanho dessa interseção será 3.

conseguiram uma redução próxima de 20%.

Em [Costa et al., 2020], os autores apresentam um novo procedimento de redução para o problema. Além disso, os autores mostram que esse procedimento domina o procedimento apresentado em Bogue et al. [2014]. Esses procedimentos de redução podem ser usados como pré-processamento do problema ou acoplados a um algoritmo de branch-and-bound combinatório. O novo procedimento de redução mostrou-se bem mais efetivo como procedimento de pré-processamento e o procedimento proposto por Bogue et al. [2014] mostrou-se mais efetivo quando utilizado em cada nó da árvore de branch-and-bound.

Neste artigo, apresentaremos um novo procedimento de redução do tamanho da entrada e uma estratégia para averiguar de forma mais forte e eficiente se um vértice pode ser removido, que denominamos de *método de redução*. Esse *método de redução* usa internamente qualquer procedimento de redução. Também mostraremos como usamos esses procedimentos de redução como uma forma de testar a otimalidade de uma determinada solução viável. Faremos exaustivos experimentos computacionais para averiguar a eficácia de todos os procedimentos e do *método de redução*.

Este artigo está organizado da seguinte forma. Na Seção 2 descrevemos os procedimentos de redução já existentes na literatura. Na Seção 3, apresentamos um novo procedimento de redução baseado na enumeração de cliques. Na Seção 4, apresentamos a nossa estratégia de redução que pode ser combinada com um procedimento de redução. Na Seção 5, relatamos os resultados dos testes computacionais realizados. E, finalmente, na seção 6, as conclusões e considerações finais são apresentadas.

2. Procedimentos de Redução

Nesta seção, apresentaremos os procedimentos de remoção de nós de $G = (L \cup R, E)$. No problema `kMIS`, a regra que permite a remoção de um vértice de L diferirá da regra de remoção de um vértice de R. As regras de remoção devem considerar o valor do parâmetro k e o valor de uma solução viável conhecida λ.

O primeiro procedimento desse tipo para o `kMIS` foi apresentado por Bogue et al. [2014]. Os autores apresentaram dois testes, um para remover vértice de L e outro para vértice de R. Como a remoção de vértices de L influência o outro teste e vice-versa, eles são aplicados em um procedimento iterativo, até que nenhum vértice seja removido, como veremos mais adiante.

O procedimento de Bogue et al. [2014] (**PRBogue**) procura encontrar vértices que não podem estar em uma solução viável de custo maior ou igual λ. Aqui iremos fazer uma pequena mudança nesse procedimento, para que ele procure vértices que não podem estar em uma solução

viável de custo **maior que** λ. Essa pequena mudança nos possibilita utilizar todos os procedimentos desta seção para avaliar a otimalidade de uma solução viável, como iremos detalhar na Seção 5.2.

Seja $\mathcal{L}_u(\lambda) = \{v \in L \setminus \{u\} \colon |N_G(u) \cap N_G(v)| \leq \lambda\}, \forall u \in L$ e $\mathcal{R}_u(t) = \{v \in R \setminus \{u\} \colon |N_G(u) \cap N_G(v)| < t\}, \forall u \in R$, onde λ é o valor de uma solução viável para o problema e $t = k$. O procedimento proposto por Bogue et al. [2014] é realizado da seguinte forma:

1. Todo vértice $u \in L$, onde $d_G(u) \leq \lambda$ ou $|L| - |\mathcal{L}_u(\lambda)| < k$, é removido de G.
2. Todo vértice $v \in R$, onde $d_G(v) < k$ ou $|R| - |\mathcal{R}_v(k)| \leq \lambda$, é removido de G.
3. Repita os passos 1 e 2 enquanto for possível reduzir o tamanho da instância.

Em [Costa et al., 2020], os autores apresentaram um outro procedimento de redução para o `kMIS`, que aqui vamos chama-lo de **Procedimento de Redução por Checagem Dupla** (**PRChDupla**), pois os testes definidos para remover um vértice não considera apenas a sua vizinhança, mas também a vizinhança dos seus vizinhos. Os autores provaram que o **PRChDupla** domina o **PRBogue**, ou seja, todo vértice removido pelo **PRBogue** também é removido pelo **PRChDupla**.

De forma semelhante como fizemos com o procedimento **PRBogue**, iremos fazer uma pequena mudança no **PRChDupla** para que ele remova vértices que não podem estar em uma solução viável de custo maior que λ.

Seja $\overline{\mathcal{L}_u}(\lambda) = L \setminus (\mathcal{L}_u(\lambda) \cup \{u\}), \forall u \in L$ e $\overline{\mathcal{R}_u}(t) = R \setminus (\mathcal{R}_u(t) \cup \{u\}), \forall u \in R$.

Definimos o conjunto $C_L(u) = \{v \in N_G(u) \colon |\overline{\mathcal{L}_u}(\lambda) \cap N_G(v)| \geq k - 1\}, \forall u \in L$ como o subconjunto de vértices em $N_G(u) \subseteq R$, tal que pertença também a vizinhança de pelo menos $k-1$ vértices de $\overline{\mathcal{L}_u}(\lambda)$. De forma análoga, definimos o conjunto $C_R(v) = \{u \in N_G(v) \colon |\overline{\mathcal{R}_v}(t) \cap N_G(u)| \geq \lambda\}$ como o subconjunto de vértices em $N_G(v) \subseteq L$, tal que pertença também a vizinhança de pelo menos λ vértices de $\overline{\mathcal{R}_v}(t)$. O procedimento proposto por [Costa et al., 2020] é realizado da seguinte forma:

1. Todo vértice $u \in L$, onde $|C_L(u)| \leq \lambda$, é removido de G.
2. Todo vértice $v \in R$, onde $|C_R(v)| < k$, é removido de G.
3. Repita os passos 1 e 2 enquanto for possível reduzir o tamanho da instância.

3. Procedimento de Redução por Cliques

Nesta seção, propomos um procedimento de redução que utiliza o fato que toda solução viável induz um grafo bipartido completo entre os k vértices de L da solução e os vértices de R da interseção, como podemos ver na Figura 2. Na Figura 3, temos um novo grafo, que denominamos de $H_L(\lambda)$, onde os vértices representam os vértices de L e as arestas conectam vértices de L cuja interseção das suas vizinhanças em G tiver tamanho pelo menos λ. Qualquer solução viável de custo maior ou igual a λ em G induz uma clique em $H_L(\lambda)$ de tamanho k, portanto, qualquer vértice de L que não está contido em uma clique de tamanho pelo menos k em $H_L(\lambda)$, pode ser removido de G.

Assim como nos demais procedimentos, estamos interessados em remover vértices que não podem estar em uma solução de custo melhor que λ, dessa forma, utilizamos o grafo $H_L(\lambda+1)$ que denominamos de G_L. Seja $G_L = (V, E)$ um grafo simples não direcionado, onde $V(G_L) = L$ e $E(G_L) = \{uv \colon \forall u \in L$ e $v \in \overline{\mathcal{L}_u}(\lambda)\}$. Portanto, se o subgrafo formado pelo vértice u juntamente com os vértices de $\overline{\mathcal{L}_u}(\lambda)$ não contém uma clique de tamanho pelo menos k, então u pode ser removido de G.

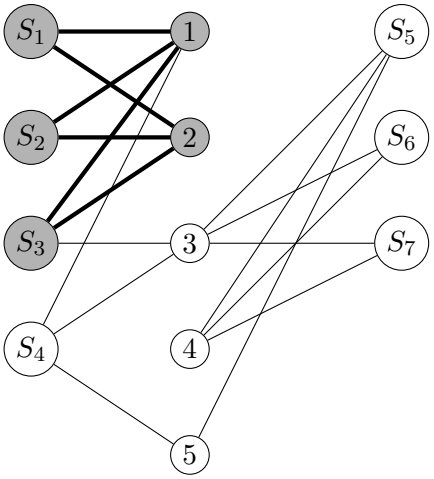


Figura 2: Neste exemplo, temos uma instância do problema para $k = 3$. Em destaque, temos uma solução viável com valor $\lambda = 2$.

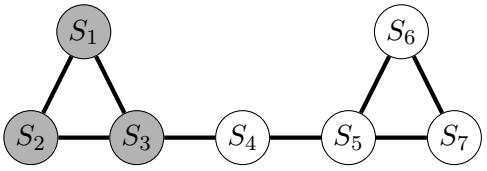


Figura 3: Uma versão do grafo $H_L(\lambda)$ onde o conjunto de vértices é formado pelos vértices de L e dois vértices são adjacentes se a interseção das suas vizinhanças tem tamanho pele menos λ

De maneira similar, podemos definir o grafo $G_R = (V, E)$, onde $V(G_R) = R$ e $E(G_R) = \{uv : \forall u \in R \text{ e } v \in \overline{\mathcal{R}_u}(t)\}$, com $t = k$. Deste modo, se o subgrafo formado pelo vértice u juntamente com os vértices de $\overline{\mathcal{R}_u}(t)$ não contém uma clique de tamanho maior que λ, então u pode ser removido de G.

Este novo procedimento de redução, que denominamos de **Procedimento de Redução por Cliques** (**PRCliques**), é realizado da seguinte forma:

1. Todo vértice $u \in V(G_L)$ que não está contido em nenhuma clique de tamanho pelo menos k em G_L, é removido de G_L e de G;

2. Todo vértice $u \in V(G_R)$ que não está contido em nenhuma clique de tamanho maior que λ em G_R, é removido de G_R e de G.

3. Repita os passos 1 e 2 enquanto for possível reduzir o tamanho da instância.

A seguir, é exibido a prova da dominância do **PRCliques** em relação ao **PRBogue**.

Proposition 1. *Seja $u \in L$, um inteiros k e $\lambda > 0$. Se $d_G(u) \leq \lambda$ ou $|L| - |\mathcal{L}_u(\lambda)| < k$, então u não pertence a um clique de tamanho pelo menos k no grafo G_L.*

Demonstração. Pela própria definição de $\overline{\mathcal{L}_u}(\lambda)$, se $d_G(u) \leq \lambda$ então $\overline{\mathcal{L}_u}(\lambda) = \emptyset$. Logo, $d_{G_L}(u) = 0$ e u não pode pertencer a uma clique de tamanho k

Suponha por absurdo, que u pertence a uma clique de tamanho pelo menos k em G_L, ou seja, $|\overline{\mathcal{L}_u}(\lambda)| \geq k - 1$. Como $|L| = |\mathcal{L}_u(\lambda)| + |\overline{\mathcal{L}_u}(\lambda)| + 1$, logo $|\overline{\mathcal{L}_u}(\lambda)| = |L| - |\mathcal{L}_u(\lambda)| - 1$. Portanto, $|L| - |\mathcal{L}_u(\lambda)| - 1 \geq k - 1 \implies |L| - |\mathcal{L}_u(\lambda)| \geq k$, absurdo, pois $|L| - |\mathcal{L}_u(\lambda)| < k$. □

Proposition 2. *Seja $v \in R$, um inteiros k e $\lambda > 0$. Se $d_G(v) < k$ ou $|R| - |\mathcal{R}_v(k)| \leq \lambda$, então v não pertence a um clique de tamanho maior que λ no grafo G_R.*

A demonstração da Proposição 2 é análoga a demonstração da Proposição 1.

A geração de todas as cliques maximais com tamanho maior que um certo limite pode ser alcançada utilizando o algoritmo apresentado em Tomita et al. [2006]. Neste artigo, os autores apresentam um algoritmo que gera todas as cliques maximais de um grafo não direcionado utilizando as regras de poda do algoritmo de Bron-Kerbosch [Bron e Kerbosch, 1973] com complexidade de pior caso de $\mathcal{O}(3^{\frac{n}{3}})$ para um grafo com n vértices. Testes computacionais reveleram que o algoritmo roda muito rápido na prática.

4. Método de Redução por Contração de Vértices

Nesta seção, apresentaremos uma estratégia de redução que começa realizando uma operação de contração de vértices e em seguida verifica se o vértice contraído pode ser removido. Note que se o vértice contraído pode ser removido, então os dois vértices contraídos não podem estar na mesma solução.

Observe que todos os procedimentos de redução apresentados até agora, possuem um teste específico para verificar se um vértice de L (ou R) pode ser removido. Utilizaremos esses testes específicos como critério para remoção do nosso vértice contraído. Por exemplo, no procedimento de redução do Bogue, o critério de remoção de um vértice de u de L é:

- se $d_G(u) \leq \lambda$ ou $|L| - |\mathcal{L}_u(\lambda)| < k - 1$, então u pode ser removido de G.

Já no método da Checagem Dupla, o critério de remoção de um vértice de u de L é:

- se $|\{v \in N_G(u) \colon |\overline{\mathcal{L}_u}(\lambda) \cap N_G(v)| \geq k - 2\}| \leq \lambda$, então u pode ser removido de G.

Veja que nos dois casos, algumas adaptações são necessárias para utilizar como um teste de remoção.

Dado quaisquer dois vértices u e v de G, a operação de contração dos vértices u e v deleta os vértices u e v e adiciona um vértice c_{uv} que é adjacente a todos os vértices adjacentes a u e v, simultaneamente. O grafo resultante deste processo será denotado por $G \setminus \{u, v\}$. Em seguida, aplicamos um teste de redução no vértice c_{uv} para verificar se ele pode ser removido. Em outras palavras, verificamos se é possível gerar alguma solução viável com valor maior que λ, tal que u e v estejam juntos nessa solução.

Seja $teste(Criterio, u)$ uma função que devolve $true$ se o vértice u pode ser removido pela critério de redução $Criterio$ e, $false$, caso contrário. Definimos então $\beta_u^L(\lambda) = \{v \in L \setminus \{u\} \colon teste(Criterio, c_{uv})\}, \forall u \in L$. Portanto, se $|L| - \beta_u^L(\lambda) < k$, então u pode ser removido de G.

De forma análoga podemos fazer o mesmo para os vértices em R. Seja $\beta_u^R(t) = \{v \in R \setminus \{u\} \colon teste(Criterio, c_{uv}) = true\}, \forall u \in R$. Se $|R| - \beta_u^R(t) \leq \lambda$, então u pode ser removido de G.

Esse método de redução, que chamaremos de **Método de Redução por Contração de Vértices** (**MTDContracao**), é realizado da seguinte forma:

1. Todo vértice $u \in L$, onde $|L| - \beta_u^L(\lambda) < k$, é removido de G.
2. Todo vértice $v \in R$, onde $|R| - \beta_u^R(k) \leq \lambda$, é removido de G.
3. Repita os passos 1 e 2 enquanto for possível reduzir o tamanho da instância.

A prova de dominância é feita de acordo com o critério de redução. Por exemplo, se utilizarmos o **PRChDupla**, então o **MTDContracao** domina o **PRChDupla**. A prova é fácil e direta, pois se o **PRChDupla** indica que u deve ser removido, então qualquer vértice que for gerado por contração junto com u, também será removido pela **MTDContracao** com o critério da **PRChDupla**.

5. Testes Computacionais e Resultados

Nesta seção, apresentamos os resultados computacionais feitos neste trabalho, além das instâncias utilizadas nos testes computacionais e ambiente computacional.

5.1. Instâncias e Ambiente Computacional

As instâncias consideradas neste trabalho foram as instâncias criadas em [Costa et al., 2020], que são grafos bipartites gerados de maneira aleatória usando o modelo de [Gilbert, 1959]. De forma resumida, são 216 instâncias (270, mas apenas 216 são viáveis) de tamanhos 40, 60, 80, 100, 140, 180, 200, 240, 280 e 300 vértices, diferenciando-se pela densidade e valor de k. Essas instâncias são divididas em 9 classes de acordo com essas propriedades, como podemos ver na Tabela 1. Além disso, temos instâncias balanceadas ($|L| = |R|$) e desbalanceadas ($|L| > |R|$ e $|L| < |R|$). Na Tabela 1, para cada classe, entre parenteses temos a quantidade total de instâncias viáveis.

Tabela 1: Classes de Instâncias por Propriedade [Costa et al., 2020]

k \ Densi.	Baixa	Média	Alta
Baixo	$C_1(30)$	$C_4(30)$	$C_7(30)$
Médio	$C_2(17)$	$C_5(30)$	$C_8(30)$
Alto	$C_3(0)$	$C_6(19)$	$C_9(30)$

Neste trabalho, os subconjuntos são representados por meio de strings de bits, desta forma, todas as operações sobre os subconjuntos em nossos algoritmos são feitas por meio de paralelismo de bits. Em [Costa et al., 2018] e [Dias et al., 2020], os autores relataram melhoria considerável no tempo computacional quando se utiliza strings de bits.

Os algoritmos foram implementados na linguagem C++ e executados em um computador Intel Xeon, com 2.30 GHz, 64 GBytes de memória RAM e usando o CentOS 7.0. Todos os algoritmos foram executados com 1 thread, com limite de memória de 8 GBytes e limitação de tempo execução de 1800 segundos. Todas as instâncias, assim como os códigos fontes desse trabalho encontram-se no repositório do GitHub no link: https://github.com/fabiocsd/kMIS.

5.2. Testes de Redução

Todos os procedimentos de redução foram definidos com a finalidade de remover vértice que não pode estar em uma solução viável de valor melhor que λ. Portanto, no final, se $|L| < k$ ou $|R| \leq \lambda$, então λ é o valor ótimo. Logo, os procedimentos de redução podem ser utilizados para provar a otimalidade de uma determinada solução. Todos os procedimentos foram implementados com esse critério de parada adicional.

Como valor de λ, usamos a melhor solução encontrada quando executamos o algoritmo VNS Reativo de [Dias et al., 2020] e o Grasp de [Casado et al., 2022], os dois algoritmos heurísticos estado da arte do problema. Iremos comparar os procedimentos em relação a quantidade de vértices removidos, a quantidade de instâncias com otimalidade provada e o tempo computacional.

Para computar os vértices removidos, somente iremos considerar nas instâncias que todos os procedimentos que estão sendo comparados não conseguiram provar a otimalidade de λ. Isso se faz necessário, pois em algumas instâncias, um procedimento prova a otimalidade removendo poucos vértices, enquanto outro também prova, mas removendo mais vértices. Por exemplo, na instância da classe C_6 com $|L| = 160$, $|R| = 200$, $k = 112$ e $\lambda = 1$, o **PRBogue** não remove nenhum vértice em L, mas remove 199 vértices de R garantindo a otimalidade de λ. Já o **PRCh-Dupla** prova a otimalidade removendo apenas 49 vértices de L ($160 - 49 < 112$). Observe que no final, **PRBogue** removeu mais vértices.

Nossa primeira avaliação será entre os procedimentos **PRBogue**, **PRChDupla** e **PRCliques**. Na Tabela 2 mostramos o número de vértices removidos e na Tabela 3 o número de ótimos provados, por classe de instância.

Inicialmente, destacamos que nas classes C_2 e C_6, classes de densidades baixa e média, respectivamente, e com k médio e alto, respectivamente, todos os procedimentos conseguiram mostrar que λ é o ótimo.

Nas classes C_8 e C_9 classes de densidade alta e k médio e, densidade alta e k alto, respectivamente, nenhum procedimento conseguiu remover algum vértice. Em [Costa et al., 2020], os autores destacam que as instâncias dessas classes foram as que mais o algoritmo exato encontrou dificuldade na sua resolução. Os resultados dos procedimentos nessas classes reforçam a suspeita que essas sejam as instâncias mais difíceis.

Tabela 2: Número de vértices removidos

Classe	**PRBogue**	**PRChDupla**	**PRCliques**
C_1	29	**74**	39
C_4	31	**44**	31
C_5	592	**776**	596
C_7	40	**47**	40
Total	692	**941**	706

Tabela 3: Número de ótimos

Classe	**PRBogue**	**PRChDupla**	**PRCliques**
C_1	1	2	1
C_2	17	17	17
C_6	19	19	19
C_7	4	4	4
Total	41	**42**	41

O **PRChDupla** conseguiu remover cerca de 36% a mais que o **PRBogue** e 33% a mais que o **PRCliques**. A quantidade de ótimos foram praticamente equivalentes. O **PRCliques** foi ligeiramente melhor que o **PRBogue**, removendo cerca de 2% a mais de vértices e, com a mesma quantidade de instâncias ótimas. Em contra-partida, o tempo médio do **PRBogue** foi de 0,0073s, enquanto do **PRChDupla** foi 0,1915s e do **PRCliques** foi de 2,7426s. Como era de se esperar, o tempo do **PRCliques** foi muito maior que dos demais. Observe que apesar do tempo médio do **PRChDupla** ser 2.523% maior que o do **PRBogue**, o valor absoluto não passar de 0,2s, justificando seu uso em detrimento do **PRBogue**.

Analisando os resultados nas instâncias de densidades baixa e média, classes C_1, C_2, C_4, C_5 e C_6, observamos que a medida que k aumenta, elas se tornam mais fáceis para os procedimentos. Mas comportamento contrário ocorre nas classes de densidade alta, C_7, C_8 e C_9. A medida que k aumenta, elas se tornam mais difíceis.

5.2.1. Método de Redução por Contração de Vértices

Como o **MTDContracao** usa internamente um dos outros três procedimentos, e como os resultados do primeiro experimento computacional mostrou que o **PRCliques** se torna computacionalmente inviável, fizemos duas execuções do **MTDContracao**, uma usando o **PRBogue** e a outra usando o **PRChDupla**. Nas Tabelas 4 e 5 temos os resultados desde experimento.

Primeiramente, queremos fazer uma análise dos resultados das Tabelas 3 e 5. Os procedimentos conseguiram provar a otimalidade de 42 instâncias. Já o **MTDContracao** provou de 103 instâncias, um aumento de cerca 145%, mostrando a eficiência desse método. Enquanto os procedimentos provaram o ótimo de 3 instâncias da classe C_1, nenhuma de C_4 e C_5, o **MTDContracao** provou de todas as instâncias das classes C_1 e C_5 e 4 da C_4. Ele também conseguiu remover um vértice de uma instância da classe C_9 (decidimos não incluir na Tabela 4).

O **MTDContracao+PRChDupla** removeu mais vértices que o **MTDContracao+PRBogue**, cerca de 20% a mais, e a quantidade de soluções ótimas foi bastante próxima com uma leve vantagem para o **MTDContracao+PRChDupla**. No tempo computacional, o **MTDContracao+PRBogue**

Tabela 4: Número de vértices removidos

Classe	**MTDContracao +PRBogue**	**MTDContracao +PRChDupla**
C_4	4	**8**
C_5	61	**70**
C_7	82	**98**
Total	147	**177**

Tabela 5: Número de ótimos

Classe	**MTDContracao +PRBogue**	**MTDContracao +PRChDupla**
C_1	30	30
C_2	17	17
C_4	3	4
C_5	28	29
C_6	19	19
C_7	4	4
Total	101	**103**

teve tempo médio de 0,4867s e o **MTDContracao+PRChDupla** teve 25,2845s. Por esse resultado dos tempos computacionais, concluirmos que a maior eficiência na remoção obtida pelo **MTDContracao+PRChDupla** não justifica seu uso. Portanto, o **PRBogue** é o melhor procedimento atualmente para se aplicar junto ao **MTDContracao**.

Pedimos um cuidado do leitor para não comparar as Tabelas 2 e 4, pois como estamos computando apenas as remoções em instâncias que nenhum procedimento (usado na comparação) provou a otimalidade e como o **MTDContracao** é mais eficiente nesse quesito, então serão computados menos vértices removidos.

Nossa última comparação será entre o melhor procedimento de uso individual, **PRChDupla**, com o melhor usando o método de redução, **MTDContracao+PRBogue**. A quantidade de ótimos e os tempos já foram mostrados nas tabelas anteriores, mas como a quantidade de vértices removidos depende dos procedimentos comparados, na Tabela 6 temos essa informação dessa comparação.

O **MTDContracao+PRBogue** remove cerca de 141% a mais vértices, prova a otimalidade de cerca 140% a mais, a um custo computacional de tempo médio de cerca de 154% a mais, em relação ao **PRChDupla**. Esse percentual de tempo a mais pode parecer muito, mas considerando o tempo médio absoluto de menos que 0,5s, concluirmos que o **MTDContracao+PRBogue** é o melhor procedimento de redução.

Tabela 6: Número de vértices removidos

Classe	**PRChDupla**	**MTDContracao +PRBogue**
C_4	13	**29**
C_5	23	**89**
C_7	47	**82**
Total	83	**200**

6. Conclusão

Neste trabalho, realizamos experimentos computacionais com procedimentos de redução existentes na literatura para o Problema de Máxima Interseção de k - subconjuntos (`kMIS`) e apresentamos outros procedimentos. Os procedimentos da literatura considerados foram o **PRBogue**[Bogue et al., 2014] e o **Procedimento de Redução por Checagem Dupla**[Costa et al., 2020]. Apresentamos outro procedimento, denominado de **Procedimento de Redução por Cliques** que

busca cliques em um grafo gerado a partir da informação da interseção da vizinhança dos vértices em comparação ao valor de uma solução viável. Provamos que ele domina o **PRBogue**. Como era de se esperar, o custo computacional desse novo teste foi bastante superior comparado aos demais.

Também desenvolvemos um método de redução do grafo de entada do problema. A estratégia desse método proposto utiliza qualquer teste de redução existente. Denominamos de **Método de Redução por Contração de Vértices**, pois ele realiza a operação de contração de dois vértices. Os resultados computacionais prova a eficiência desse método de redução, mostrando que a ideia é bastante promissora. Ele conseguiu remover mais vértices e mostrar a otimalidade de mais soluções de instâncias a um custo computacional aceitável. O **PRBogue** foi o procedimento de redução que mostrou-se ter o maior custo benefício de ser usado junto com o **MTDContracao**.

Como trabalho futuro pretendemos desenvolver outros procedimentos de redução e procurar por testes que visem definir se um dos vértices está em alguma solução ótima ou mesmo testes para remoção de arestas. Além de procurarmos implementações mais eficientes dos procedimentos, por usar informações de quais vértices são afetados de uma iteração para outra, ao invés de verificar todos os vértices, como foi feito na implementação atual.

Agradecimentos

Essa pesquisa foi parcialmente apoiado pela FUNCAP PS1-0186-00155.01.00/21, FUNCAP PNE 0112.00061.01.00-16, CNPq Proc. 309315/2019-0, CNPq Proc. 422912/2021-2 e Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Agradecemos ao revisor por seus comentários e sugestões que nos ajudaram a melhorar o artigo.

Referências

Acuña, V., Ferreira, C. E., Freire, A. S., e Moreno, E. (2014). Solving the maximum edge biclique packing problem on unbalanced bipartite graphs. *Discrete Applied Mathematics*, 164:2–12.

Bogue, E. T., de Souza, C. C., Xavier, E. C., e Freire, A. S. (2014). An integer programming formulation for the maximum k-subset intersection problem. In *Combinatorial Optimization*, p. 87–99, Cham. Springer International Publishing. ISBN 978-3-319-09174-7.

Bron, C. e Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577.

Casado, A., Pérez-Peló, S., Sánchez-Oro, J., e Duarte, A. (2022). A grasp algorithm with tabu search improvement for solving the maximum intersection of k-subsets problem. *Journal of Heuristics*, 28(1):121–146.

Costa, J. R. d. F., Dias, F. C. S., e de A. Tavares, W. (2018). Um algoritmo vns para o problema da máxima interseção de k-subconjuntos. In *Anais do 50 Simpósio Brasileiro de Pesquisa Operacional*. Sobrapo.

Costa, J. R. d. F., Dias, F. C. S., e de A. Tavares, W. (2020). Um algoritmo branch-and-bound para o problema da máxima interseção de k-subconjuntos. *Anais do LII Simpósio Brasileiro de Pesquisa Operacional*.

Dias, F. C., Tavares, W. A., e de Freitas Costa, J. R. (2020). Reactive vns algorithm for the maximum k-subset intersection problem. *Journal of Heuristics*, 26(6):913–941.

Gilbert, E. N. (1959). Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144.

Nussbaum, D., Pu, S., Sack, J.-R., Uno, T., e Zarrabi-Zadeh, H. (2012). Finding maximum edge bicliques in convex bipartite graphs. *Algorithmica*, 64(2):311–325.

Shieh, M.-Z., Tsai, S.-C., e Yang, M.-C. (2012). On the inapproximability of maximum intersection problems. *Information Processing Letters*, 112(19):723–727.

Tomita, E., Tanaka, A., e Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical computer science*, 363(1):28–42.

Vinterbo, S. (2002). A note on the hardness of the k-ambiguity problem. Technical report, Harvard Medical School, Boston, MA, USA.

Xavier, E. C. (2012). A note on a maximum k-subset intersection problem. *Information Processing Letters*, 112(12):471–472.