O problema da Máxima Interseção de k-Subconjuntos

Eduardo Theodoro Bogue[1], Cid Carvalho de Souza[1], Eduardo Candido Xavier[1] e Alexandre da Silva Freire[1]

[1]Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)
Campinas - SP - Brasil
{eduardotheodoro}@gmail.com, {cid, ecx, afreire}@ic.unicamp.br

Resumo

Neste artigo apresentamos três formulações de programação linear inteira e uma heurística para o Problema da Máxima Interseção de k-Subconjuntos. Trata-se de um problema $\mathcal{NP}$-difícil e que ocorre em aplicações de diversas áreas como biologia computacional e privacidade de dados. Experimentos empíricos foram conduzidos com o objetivo de avaliar a qualidade dos modelos propostos e a qualidade das soluções geradas pela heurística.

Palavras-chave: Problema da Máxima Interseção de k-Subconjuntos, Programação Linear Inteira, Algoritmos exatos e heurísticas.

Área principal: Otimização Combinatória, Teoria e Algoritmos em Grafos.

Abstract

This paper presents three integer linear programming formulations and one heuristic for the Maximum k-Subset Intersection Problem. This is a $\mathcal{NP}$-hard problem and occurs in several applications areas such as computation biology and data privacy. Empirical experiments were conducted to evaluate the quality of the proposed models and the quality of the solutions genereted by the heuristic.

Keywords: Maximum k-Subset Intersection Problem, Integer Linear Programming, Exact algorithms and heuristics.

Main Area: Combinatorial Optimization, Theory and Algorithms in Graphs.

1 Introdução

O Problema da Máxima Interseção de k-Subconjuntos (kMIS) é definido do seguinte modo: Dada uma coleção $L = \{S_1, \ldots, S_n\}$ de n subconjuntos sobre um conjunto finito de elementos $R = \{e_1, \ldots, e_m\}$, e um inteiro positivo k, o objetivo é selecionar exatamente k subconjuntos pertencentes a L, tal que a interseção destes subconjuntos seja máxima. O problema pode ser representado com um grafo bipartido onde os vértices do lado esquerdo da bipartição representam os subconjuntos de L e os vértices do lado direito os elementos de R. Uma aresta liga um vértice i associado a $S_i \in L$ a um vértice j correspondente a $e_j \in R$ se e somente se $e_j \in S_i$. Na Figura 1 é mostrado o grafo bipartido referente à instância do problema em que $k = 2$, $L = \{S_1, S_2, S_3, S_4\}$ e $R = \{1, 2, 3, 4, 5\}$, onde $S_1 = \{2, 3\}$, $S_2 = \{1, 2, 3\}$, $S_3 = \{1, 2, 3, 4, 5\}$ e $S_4 = \{2, 3, 5\}$.

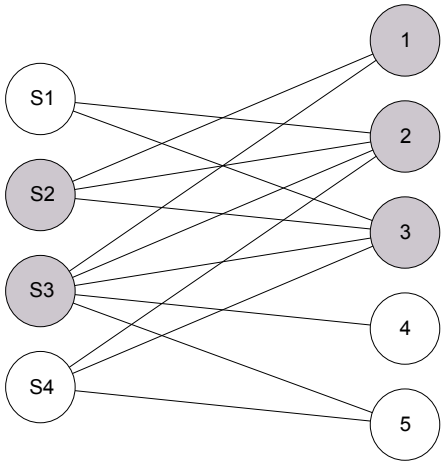

Figura 1: Para $k = 2$, a maior interseção é obtida com os subconjuntos S_2 e S_3, dada por $\{1, 2, 3\}$.

A motivação inicial para investigar o kMIS origina-se de sua aplicação na análise de dados em bioinformática. [Nussbaum et al., 2010] citam o exemplo da tecnologia de DNA Microarray, que recentemente tornou-se uma das principais ferramentas para o estudo da interação entre genes e condições de teste. Dados de Microarray são usualmente apresentados como uma matriz bidimensional em que as linhas representam genes e as colunas condições de teste. Cada entrada $[i, j]$ da matriz binária representa o nível de expressão de um dado gene i sobre uma determinada condição j. Para melhor entender processos biológicos de uma célula, biólogos geralmente tentam encontrar relações entre subconjuntos de genes e subconjutos de condições. Contudo, esta análise se torna inviável de ser realizada sem a utilização de recursos computacionais. Este problema pode ser modelado como uma instância do problema kMIS de forma que L é o conjunto de genes e R o conjunto de condições, e o objetivo é selecionar k genes que possuam o maior número de condições em comum.

Outra aplicação para o problema consiste na tarefa de encontrar um grupo de pessoas com o maior número de interesses em comum em redes sociais. Seja L o conjunto de pessoas em uma rede social e R um conjunto de interesses, o objetivo é encontrar um grupo de k pessoas em L com maior quantidade de interesses em comum.

[Vinterbo, 2002] propõe que o kMIS seja utilizado como um modelo matemático para o problema de controle de divulgação de dados. Seja L um conjunto de n pacientes em um hospital, R um conjunto de dados dos pacientes. Para assegurar que os dados divulgados não possam ser utilizados para identificar nenhum dos pacientes, é permitido revelar algum dado somente se k pacientes possuam este mesmo dado, de maneira que k seja grande o suficiente para garantir a preservação da privacidade. Com isto, o objetivo do problema é encontrar k pacientes que possuam a maior quantidade possível de dados em comum.

Os objetivos deste trabalho são: propor modelos de programação linear inteira (PLI) com o intuito de se resolver o problema de maneira exata e uma heurística. Não foi encontrado na literatura nenhum outro trabalho onde tenha sido feito um tratamento algorítmico do problema.

Este trabalho está estruturado da seguinte forma. Na Seção 2 apresentamos uma revisão da bibliografia. Na Seção 3 elencamos três modelos de programação linear inteira desenvolvidos para o problema. Na Seção 4 exibimos uma heurística gulosa desenvolvida para o problema. Na Seção 5 resumimos os resultados obtidos pelos modelos de PLI e pela heurística desenvolvida. Por último, apresentamos na Seção 6 as conclusões e trabalhos futuros.

2 Revisão Bibliográfica

Nesta seção é feita uma revisão dos trabalhos anteriores que apresentam problemas relacionados ao kMIS. Pelo que se sabe até então, foi [Vinterbo, 2002] o primeiro a citar este problema na literatura. Em relação à complexidade computacional, em [Vinterbo, 2002] e [Xavier, 2012] é demonstrado que o problema é $\mathcal{NP}$-difícil e em [Xavier, 2012], é também demonstrado que a menos que $\mathcal{P}$=$\mathcal{NP}$, o kMIS não admite algoritmo $\frac{1}{N^{\epsilon}}$-aproximado, sendo N o tamanho da entrada e ϵ uma constante. Para provar a complexidade do problema, [Xavier, 2012] realiza uma redução a partir do problema *Maximum Edge Biclique (MEB)*, provado ser $\mathcal{NP}$-difícil por [Peeters, 2003]. O problema MEB é definido do seguinte modo: dado um grafo bipartido $G = (L, R, E)$, o objetivo do problema é encontrar uma biclique C que possua cardinalidade máxima de arestas. Uma biclique $C = U_1 \cup U_2$ é um subconjunto do conjunto de vértices, tal que $U_1 \subseteq L$, $U_2 \subseteq R$ e para cada $u \in U_1, v \in U_2$ a aresta (u, v) existe em E. Em outras palavras, uma biclique é um subgrafo bipartido completo de G. Note que o problema kMIS é equivalente ao problema MEB com a restriçao adicional de que $|U_1| = k$. A Figura 2 exibe uma instância do problema MEB.

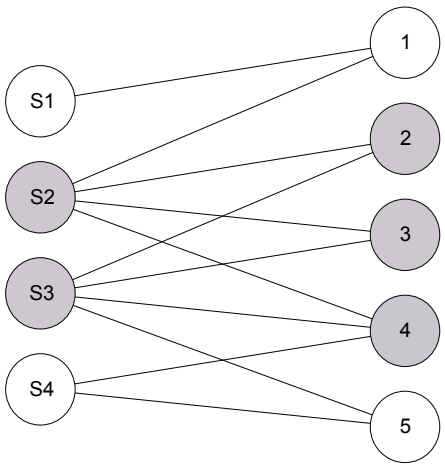

Figura 2: Instância do problema MEB em que os vértices em cinza representam a maior biclique encontrada no grafo.

[Acuña et al., 2011] propõem formulações em programação linear inteira para o problema MEB. A partir das formulações propostas são realizados experimentos computacionais utilizando instâncias provenientes de aplicações em bioinformática e instâncias geradas aleatoriamente.

[Nussbaum et al., 2010] apresentam um algoritmo para o problema MEB no caso especial onde $G = (L, R, E)$ é um grafo *convexo* em R (G é dito ser *convexo* em R se existir uma ordenação de vértices em R tal que, para qualquer vértice $v \in L$, os vértices adjacentes a v são consecutivos em R). O algoritmo apresentado possui complexidade $O(n \log^3 n \log \log n)$ em tempo e $O(n)$ em espaço, onde $n = |E|$. Em [Alexe et al., 2004], [Dias et al., 2005] e [Dias et al., 2007], os autores

apresentam algoritmos exponenciais capazes de gerar todas as bicliques de um grafo G, tornando assim possível encontrar uma solução para o problema MEB através de uma simples busca pela biclique de maior número de arestas.

3 Formulações de PLI

Nesta seção, apresentamos duas formulações para o MEB descritas em [Acuña et al., 2011], mostramos como é possível adaptá-las para o kMIS e, por último, introduzimos uma nova formulação de PLI para o kMIS.

Seja $G = (V, E)$ um grafo (L, R)-bipartido. Dado um vértice $u \in V$, denotamos por $N(u) \subset V$ o conjunto dos vértices adjacentes a u, e definimos como $\overline{N}(u) = R \setminus N(u)$, caso $u \in L$ e $\overline{N}(u) = L \setminus N(u)$ se $u \in R$. Denotamos o grau de um vértice u por $d(u) = |N(u)|$. No problema kMIS, os vértices em L correspondem aos n subconjuntos $\{S_1, \ldots, S_n\}$ e os vértices em R os elementos $\{e_1, \ldots, e_m\}$. Um elemento e_j pertence ao subconjunto S_i se existe uma aresta ligando e_j e S_i, de forma que $i \in L$ e $j \in R$, conforme mostrado no exemplo da Figura 1.

3.1 Formulação *MEB V* adaptada para o kMIS

Para a formulação *MEB V*, as seguintes variáveis são definidas:

y_u = quantidade de vértices vizinhos de u que pertencem a solução, caso u também pertença a solução, e 0 caso contrário, sendo que $u \in L$.

$$z_u = \begin{cases} 1, \text{ caso o vértice } u \text{ faça parte da solução.} \\ 0, \text{ c.c} \end{cases}$$

Com base nas definições exibidas, tem-se a seguinte formulação de PLI:

$$\text{Maximize} \quad \frac{1}{k} \sum_{u \in L} y_u$$

$$\begin{aligned}
\text{sujeito a} \quad & z_u + z_v \leq 1, && \forall u \in L \text{ e } \forall v \in \overline{N}(u) && (1)\\
& y_u \leq d(u) z_u, && \forall u \in L && (2)\\
& y_u \leq \sum_{v \in N(u)} z_v, && \forall u \in L && (3)\\
& \sum_{u \in L} z_u = k && && (4)\\
& y_u \geq 0, && \forall u \in L && (5)\\
& z_u \in \{0, 1\}, && \forall u \in V && (6)
\end{aligned}$$

A restrição 1 assegura que caso um vértice $u \in L$ seja escolhido, nenhum vértice de $v \in R$ que não seja vizinho de u pode ser selecionado e vice-versa. A restrição 2 força que a quantidade máxima de vértices vizinhos de $u \in L$ pertencentes a solução seja $d(u)$, caso u tenha sido selecionado, e 0 caso contrário. A restrição 3 previne que y_u seja maior que a quantidade de vértices vizinhos de u que foram escolhidos. A restrição 4 foi adicionada à formulação original e assegura que k vértices em L sejam escolhidos para a solução; em outras palavras, a quantidade de subconjuntos escolhidos no problema kMIS é igual a k, em concordância com a especificação do kMIS. As restrições 5 e 6 são restrições de integralidade. Na formulação MEB V, existem $O(|V|)$ variáveis e $O(|L||R|)$ restrições. A formulação MEB L descrita a seguir corresponde a uma modificação da formulação MEB V para reduzirmos a quantidade de variáveis binárias para $O(|L|)$.

3.2 Formulação *MEB L* adaptada para o kMIS

Para a variante da formulação *MEB L* utilizada para o problema kMIS, são utilizadas as mesmas variáveis descritas na formulação mostrada em 3.1, exceto que as variáveis z são definidas apenas para os vértices de L e as variáveis y apenas para os vértices em R. Note que na formulação MEB V as variáveis y eram definidas apenas para L.

$$\text{Maximize} \quad \frac{1}{k}\sum_{v\in R} y_v$$

$$\begin{aligned}
\text{sujeito a} \quad & y_v \leq \sum_{u\in N(v)} z_u, && \forall v \in R && (7)\\
& y_v \leq (1-z_u)d(v), && \forall u \in L, \forall v \in \overline{N}(u) && (8)\\
& \sum_{u\in L} z_u = k && && (9)\\
& y_v \geq 0, && \forall v \in R && (10)\\
& z_u \in \{0,1\}, && \forall u \in L && (11)
\end{aligned}$$

A restrição 7 força que y_v não seja maior que a quantidade de vértices vizinhos de v que foram escolhidos. A restrição 8 assegura que caso um vértice $u \in L$ seja escolhido, nenhum vértice $v \in R$ que não seja vizinho de u possa ser escolhido. A restrição 9 corresponde à mesma restrição da formulação da Seção 3.1, forçando que k vértices em L sejam escolhidos para a solução, satisfazendo assim a restrição para o problema kMIS em que k subconjuntos devem ser selecionados. As restrições 10 e 11 são restrições de integralidade.

3.3 kInter

Nesta seção apresentamos uma nova formulação para o kMIS. As variáveis utilizadas na formulação estão definidas abaixo:

$$x_i = \begin{cases} 1, \text{ caso } e_i \text{ faça parte da solução.} \\ 0, \text{ c.c} \end{cases}$$

$$y_j = \begin{cases} 1, \text{ caso } S_j \text{ seja um dos } k \text{ subconjuntos da solução.} \\ 0, \text{ c.c} \end{cases}$$

Com isto, o modelo fica sendo:

$$\text{Maximize} \quad \sum_{i\in R} x_i$$

$$\begin{aligned}
\text{sujeito a} \quad & \sum_{j\in L} y_j = k && && (12)\\
& x_i + y_j \leq 1, && \forall j \in L, \forall i \in \overline{N}(j) && (13)\\
& x_i \in \{0,1\}, && \forall i \in R && (14)\\
& y_j \in \{0,1\}, && \forall j \in L && (15)
\end{aligned}$$

A restrição 12 força que k subconjuntos sejam escolhidos. A restrição 13 assegura que e_i não pode estar na interseção caso S_j tenha sido escolhido quando $e_i \notin S_j$. As restrições 14 e 15 forçam integralidade das variáveis.

4 Heurística

Nesta seção apresentamos uma heurística gulosa desenvolvida para o kMIS. A ideia do algoritmo guloso consiste em a cada iteração, adicionar a solução o subconjunto S cuja interseção de todos os subconjuntos já adicionados a solução e S seja máxima. Inicialmente, apenas o subconjunto que possui o maior número de elementos faz parte da solução, sendo necessário a realização do procedimento de adicionar subconjuntos até que k subconjuntos estejam na solução, como apresentado no Algoritmo 1.

Algoritmo 1: Heurística Gulosa para o kMIS.

Data: **Grafo G (L,R)-bipartido**, **inteiro** k
Result: Limitante primal para o kMIS

```
início
    Seja E = R.
    Solucao = ∅.
    enquanto |Solucao| ≤ k faça
        Selecionar um subconjunto de S ∈ L de maneira que S ∉ Solucao e que E ∩ S seja
        máxima.
        E = E ∩ S.
        Solucao = Solucao ∪ S.
    fim
    return Solucao
fim
```

5 Avaliação Experimental

Para a realizaçao da avaliação experimental foram geradas instâncias aleatórias utilizando o modelo de grafo descrito em [Erdos and Renyi, 1960], onde o grafo bipartido é construído de forma que cada aresta entre um subconjunto e um elemento tem uma probabilidade independente p de ser adicionada.

As instâncias geradas foram divididas em grupos, de acordo com a densidade e a quantidade k de subconjuntos que devem ser selecionados. A divisão de k foi estabelecida de modo que k é considerado baixo quando $0.1|L| \leq k \leq 0.3|L|$, médio quando $0.4|L| \leq k \leq 0.6|L|$ e alto quando $0.7|L| \leq k \leq 0.9|L|$, tal que $|L|$ representa o número de subconjuntos do grafo de entrada. Para a densidade do grafo, foi estabelecido que o grafo é considerado ter densidade baixa quando $0.1|E| \leq densidade \leq 0.3|E|$, média quando $0.4|E| \leq densidade \leq 0.6|E|$ e alta quando $0.7|E| \leq densidade \leq 0.9|E|$, tal que $|E|$ representa o número total de arestas do grafo bipartido. Os experimentos foram realizados utilizando grafos bipartidos balanceados (em que cada lado da bipartição possui o mesmo número de vértices) com 20, 30, 40, 50 e 60 vértices em cada lado da bipartição para todas as classes de instâncias descritas, totalizando 1350 instâncias. No total, realizando a combinação dos grupos de densidade e k, as seguintes classes de instâncias foram geradas:

Classe	Densidade	k
C_1	baixa	baixo
C_2	baixa	médio
C_3	baixa	alto
C_4	média	baixo
C_5	média	médio

Classe	Densidade	k
C_6	média	alto
C_7	alta	baixo
C_8	alta	médio
C_9	alta	alto

onde em cada classe foi gerado um conjunto de 30 instâncias aleatórias utilizando os parâmetros definidos pela classe. Quando apresentando os resultados, para cada classe e tamanho de grafo, apresentamos a média de valores obtidos para as 30 instâncias. Acredita-se que o agrupamento das instâncias em classes não prejudique à exposição dos resultados obtidos, visto que o comportamento de cada algoritmo é semelhante para instâncias de uma mesma classe.

Para cada instância foi estabelecido um tempo limite de 30 minutos (1800 segundos) para sua execução. Em todas as instâncias foi realizado um pré-processamento de modo a remover os elementos que não estivessem contidos em ao menos k subconjuntos, visto que estes elementos não poderiam fazer parte da solução ótima do problema. Para a realização da avaliação experimental, utilizamos uma máquina Intel(R) Xeon(R) @2.40GHz e 8GB de memória RAM.

Nas Tabelas 1 e 2 são apresentados os resultados dos modelos de PLI. Dessa forma, são exibidos a classe, o *gap* de integralidade (em %) entre a solução da relaxação linear e a melhor solução primal (Gap_I), o *gap* de dualidade (em %) entre a melhor solução dual e a melhor solução primal encontrada (Gap_D) e o tempo de execução (em segundos) para os três modelos de PLI ($Tempo$), além da quantidade de subconjuntos ($|L|$) e de elementos ($|R|$) da instância. Denotamos em negrito os menores *gaps* de integralidade e dualidade e o menor tempo de execução.

Classe	kInter			MEB V			MEB L				
	Gap_I	Gap_D	$Tempo$	Gap_I	Gap_D	$Tempo$	Gap_I	Gap_D	$Tempo$	$\|L\|$	$\|R\|$
C_1	**15.18**	**0.00**	0.06	19.83	**0.00**	0.12	74.80	**0.00**	**0.03**	20	20
	356.92	**0.00**	0.27	**122.37**	**0.00**	0.63	184.51	**0.00**	**0.20**	30	30
	809.41	**0.00**	**0.64**	**275.40**	**0.00**	3.54	374.97	**0.00**	1.31	40	40
	643.97	**0.00**	**0.97**	**232.88**	**0.00**	4.95	355.63	**0.00**	3.32	50	50
	1444.58	**0.00**	**4.03**	**500.67**	**0.00**	214.71	604.41	**0.00**	24.82	60	60
C_2	**0.00**	**0.00**	**0.00**	1.24	**0.00**	**0.00**	4.90	**0.00**	**0.00**	20	20
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	30	30
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	40	40
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	50	50
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	60	60
C_3	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	20	20
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	30	30
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	40	40
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	50	50
	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	60	60
C_4	95.32	**0.00**	**0.08**	**70.74**	**0.00**	0.45	81.94	**0.00**	0.14	20	20
	334.26	**0.00**	**0.53**	**203.50**	**0.00**	4.44	219.00	**0.00**	1.58	30	30
	523.65	**0.00**	**1.96**	**324.99**	**0.00**	23.77	342.34	**0.00**	15.27	40	40
	745.25	**0.00**	**7.54**	**487.00**	**0.00**	128.91	512.87	**0.00**	164.66	50	50
	936.56	**0.00**	**48.90**	**599.87**	1.48	699.83	630.24	9.06	870.20	60	60
C_5	**12.94**	**0.00**	**0.05**	84.09	**0.00**	0.31	150.23	**0.00**	0.07	20	20
	196.59	**0.00**	**0.13**	314.91	**0.00**	2.12	426.56	**0.00**	1.10	30	30
	589.03	**0.00**	**0.40**	661.05	**0.00**	60.49	843.53	**0.00**	12.37	40	40
	657.34	**0.00**	**0.74**	719.97	**0.00**	482.89	875.41	**0.00**	69.95	50	50
	965.40	**0.00**	**7.41**	1039.67	**0.00**	597.08	1171.91	**0.00**	520.57	60	60

Tabela 1: Resultados dos modelos de PLI

Analizando as Tabelas 1 e 2, podemos observar que a formulação kInter resolveu de forma ótima todas as instâncias das 9 classes, visto que os valores dos *gaps* de dualidade são nulos. Entre os modelos MEB V e MEB L, nota-se que o modelo MEB L obteve em média um *gap* de dualidade inferior ao modelo MEB V.

Com relação ao tempo de execução dos modelos, o modelo kInter obteve os melhores resultados em todas as instâncias executadas, sendo em média aproximadamente 34 vezes mais rápido que o modelo MEB L e 41 vezes mais rápido que o modelo MEB V.

Além disso, com excessão das classes C_1 e C_4, o modelo kInter obteve os menores *gaps* de integralidade, o que mostra que em geral o modelo é mais apertado que os demais. Observa-se também que para as instâncias onde o valor de k é baixo, todos os modelos de PLI obtiveram os piores resultados em termos de tempo de execução, como pode ser observado nas classes C_1, C_4 e C_7, o que sugere que quanto maior a k-interseção máxima obtida, mais difícil se torna o

Classe	kInter			MEB V			MEB L				
	Gap_I	Gap_D	$Tempo$	Gap_I	Gap_D	$Tempo$	Gap_I	Gap_D	$Tempo$	$\|L\|$	$\|R\|$
C_6	**0.00**	**0.00**	**0.00**	0.71	**0.00**	0.01	9.65	**0.00**	**0.00**	20	20
	0.00	**0.00**	**0.00**	0.32	**0.00**	**0.00**	1.19	**0.00**	**0.00**	30	30
	0.00	**0.00**	**0.00**	0.36	**0.00**	**0.00**	1.61	**0.00**	**0.00**	40	40
	0.00	**0.00**	**0.00**	0.38	**0.00**	0.06	2.86	**0.00**	**0.00**	50	50
	0.00	**0.00**	**0.00**	0.23	**0.00**	0.01	0.82	**0.00**	**0.00**	60	60
C_7	**6.47**	**0.00**	**0.03**	19.56	**0.00**	0.29	20.48	**0.00**	0.06	20	20
	39.56	**0.00**	**0.24**	49.41	**0.00**	6.86	48.19	**0.00**	1.23	30	30
	70.51	**0.00**	**1.14**	78.87	**0.00**	287.65	78.63	**0.00**	25.82	40	40
	116.55	**0.00**	**8.68**	124.73	19.28	1177.77	124.56	0.28	437.69	50	50
	160.28	**0.00**	**63.03**	179.04	76.05	1716.89	173.31	24.74	1380.42	60	60
C_8	**4.62**	**0.00**	**0.04**	56.94	**0.00**	0.39	71.17	**0.00**	0.12	20	20
	96.86	**0.00**	**0.16**	71.36	**0.00**	3.90	177.80	**0.00**	2.05	30	30
	235.98	**0.00**	**0.84**	357.22	**0.00**	40.99	372.84	**0.00**	60.20	40	40
	229.58	**0.00**	**1.72**	370.16	0.87	275.95	375.39	0.42	484.67	50	50
	335.84	**0.00**	**13.55**	505.40	9.74	804.53	524.94	30.71	1457.32	60	60
C_9	**0.00**	**0.00**	**0.01**	1.16	**0.00**	0.03	3.28	**0.00**	**0.01**	20	20
	6.46	**0.00**	**0.05**	44.00	**0.00**	0.23	89.56	**0.00**	0.13	30	30
	37.36	**0.00**	**0.10**	75.43	**0.00**	0.47	114.68	**0.00**	0.34	40	40
	88.89	**0.00**	**0.28**	134.95	**0.00**	5.86	173.52	**0.00**	1.59	50	50
	119.81	**0.00**	**0.48**	163.05	**0.00**	128.14	203.69	**0.00**	5.28	60	60

Tabela 2: Resultados dos modelos de PLI

Classe	Gap_{20}	Gap_{30}	Gap_{40}	Gap_{50}	Gap_{60}
C_1	26.66	37.33	11.11	45.00	33.05
C_2	3.33	6.66	0.00	0.00	0.00
C_3	0.00	0.00	0.00	0.00	0.00
C_4	4.61	27.16	25.05	25.46	24.51
C_5	17.22	35.00	31.66	29.44	34.44
C_6	13.33	0.00	0.00	0.00	3.33
C_7	1.42	3.78	6.68	9.06	10.65
C_8	3.67	21.29	25.83	24.28	28.96
C_9	5.55	22.91	30.61	36.83	31.44

Tabela 3: Resultados da heurística gulosa

problema. Podemos observar também que a quantidade de subconjuntos e elementos possui uma grande influência na dificuldade do problema, aumentando em alguns casos cerca de 8 vezes o tempo de execução do modelo PLI, como pode ser observado na classe C_7.

Para as classes C_2, C_3 e C_6, temos que para a maioria das instâncias obtemos um *gap* de integralidade baixo, visto que como os valores de k se aproximam ou são maiores que o valor da densidade do grafo, o pré-processamento consegue remover grande parte dos elementos das instâncias.

Com relação a quantidade de instâncias que foram resolvidas na otimalidade, de um total de 1350 instâncias, o modelo kInter resolveu todas as 1350 instâncias, enquanto o modelo MEB V resolveu 1298 instâncias e o modelo MEB L 1306 instâncias. Em relação a heurística gulosa desenvolvida, analisamos o *gap* (em %) entre o valor da solução da heurística e a solução ótima, obtida através dos modelos de PLI e a quantidade de soluções onde o algoritmo atingiu a solução ótima. A Tabela 3 exibe o *gap* médio das soluções geradas pela heurística e a solução ótima para todas as classes. Para cada classe, são exibidos os valores de Gap_x, que representa o *gap* entre a heurística primal e a solução ótima para grafos que possuem x subconjuntos e x elementos.

Em média, o *gap* obtido através de uma heurística gulosa simples foi de aproximadamente 23% e a quantidade de instâncias onde foi possível atingir a solução ótima do problema foi de 782, o que pode sugerir que heurísticas mais sofisticadas diminuam ainda mais o *gap* e gerem soluções muito próximas da solução ótima do problema.

6 Conclusão

Neste trabalho, foram propostos três modelos de programação linear inteira e um algoritmo guloso para o problema da Máxima Interseção de k-Subconjuntos, que até então não havia sido tratado de maneira algoritmica.

Os resultados demonstram que a formulação desenvolvida para o kMIS obteve resultados superiores as formulações adaptadas do problema MEB, contudo, os limitantes duais fornecidos pelo modelo ainda são fracos.

Com relação a heurística desenvolvida, os resultados demonstraram que o *gap* entre as soluções onde foi possível provar a otimalidade através dos modelos de PLI e a solução do algoritmo guloso foi de cerca de 23%, o que pode sugerir que algoritmos heurísticos mais sofisticados como *Greedy Randomized Adaptive Search Procedure* (GRASP) ou Busca Tabu possam diminuir ainda mais este gap, gerando soluções muito próximas a solução ótima do problema.

Além disso, seria interessante a realização de uma análise sobre o comportamento dos modelos de PLI e da heurística para grafos bipartidos desbalanceados (onde cada lado da bipartição possui um número diferente de vértices).

Por fim, um estudo poliédrico do problema pode ser promissor na busca em se obter limitantes melhores para os modelos PLI, além da possível utilização de técnicas como *branch-and-cut* e *branch-and-price* para o problema.

7 Agradecimentos

Os autores gostariam de agradecer as agências de pesquisa brasileira FAPESP (projetos 2012/08298-6 e 2012/17585-9), CAPES e CNPq pelo apoio financeiro a este trabalho.

Referências

[Acuña et al., 2011] Acuña, V., Ferreira, C., Freire, A., and Moreno, E. (2011). Solving the maximum edge biclique packing problem on unbalanced bipartite graphs. *Discrete Applied Mathematics*. (In Press).

[Alexe et al., 2004] Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P. L., and Simeone, B. (2004). Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1):11 – 21.

[Dias et al., 2005] Dias, V., de Figueiredo, C., and Szwarcfiter, J. L. (2005). Generating bicliques of a graph in lexicographic order. *Theoretical Computer Science*, 337(1-3):240 – 248.

[Dias et al., 2007] Dias, V., de Figueiredo, C., and Szwarcfiter, J. L. (2007). On the generation of bicliques of a graph. *Discrete Applied Mathematics.*, 155(14):1826–1832.

[Erdos and Renyi, 1960] Erdos, P. and Renyi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5:17–61.

[Nussbaum et al., 2010] Nussbaum, D., Pu, S., Sack, J., Uno, T., and Takeaki, H. (2010). Finding maximum edge bicliques in convex bipartite graphs. In Thai, M. and Sahni, S., editors, *Computing and Combinatorics*, volume 6196 of *Lecture Notes in Computer Science*, pages 140–149. Springer Berlin / Heidelberg.

[Peeters, 2003] Peeters, R. (2003). The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131:651 – 654.

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

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