UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

RICARDO MOLINARI DOS PRAZERES

Modelos de programação inteira para o problema de busca de motivos em redes biológicas

São Paulo
2022

RICARDO MOLINARI DOS PRAZERES

Modelos de programação inteira para o problema de busca de motivos em redes biológicas

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Orientador: Prof. Dr. Alexandre da Silva Freire

São Paulo

2022

Ficha catalográfica elaborada pela Biblioteca da Escola de Artes, Ciências e Humanidades,
com os dados inseridos pelo(a) autor(a)
Brenda Fontes Malheiros de Castro CRB 8-7012; Sandra Tokarevicz CRB 8-4936

Prazeres, Ricardo Molinari dos
Modelos de programação inteira para o problema de busca de motivos em redes biológicas / Ricardo Molinari dos Prazeres; orientador, Alexandre da Silva Freire. -- São Paulo, 2022.
81 p: il.

Dissertacao (Mestrado em Ciencias) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, 2022.
Versão corrigida

1. busca de motivos. 2. programação inteira. 3. motivo em grafos. 4. branch-and-cut. 5. redes de interação proteína-proteína. I. Freire, Alexandre da Silva, orient. II. Título.

Dissertação de autoria de Ricardo Molinari dos Prazeres, sob o título **"Modelos de programação inteira para o problema de busca de motivos em redes biológicas"**, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em ____ de _____ de _____ pela comissão julgadora constituída pelos doutores:

Prof. Dr.
Instituição
Presidente

Prof. Dr.
Instituição

Prof. Dr.
Instituição

Dedico este trabalho aos meus amados pais e à minha maravilhosa esposa.

Agradecimentos

Aos meus professores da EACH, especialmente ao meu orientador Alexandre da Silva Freire, que dedicou muito (mas muito mesmo) de seu tempo me ajudando a desenvolver este trabalho. Mostrou a importância de um texto bem escrito (espero ter chegado perto de sua expectativa) e me deu muitos conselhos referentes à programação. Tudo isso, na maioria das vezes, através de excelentes alegorias, como por exemplo quando disse que meu texto não estava claro: "se você quer me chamar para surfar, diga: 'vamos surfar?', ao invés de jogar uma sunga e uma prancha na minha cara". Esses momentos de descontração ajudaram muito. Obrigado por tudo mesmo, professor!

À minha família, especialmente aos meus pais por tamanha dedicação e esforço para que eu e meus irmãos pudéssemos ter uma boa educação; aos meus irmãos por me darem o exemplo, sendo excelentes seres humanos; ao meu filho, que me deu a força necessária justamente na reta final; e à minha maravilhosa esposa, companheira e amiga, que sempre me apoiou e se desdobrou para que eu pudesse me dedicar neste trabalho; AMO MUITO TODOS VOCÊS.

Resumo

PRAZERES, Ricardo Molinari dos. **Modelos de programação inteira para o problema de busca de motivos em redes biológicas**. 2022. 81 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2022.

Existem diversas variantes do problema de *busca de motivos* na literatura, com muitas aplicações em bioinformática. Na variante denominada busca de motivos em grafos, proposta em 2006, dado grafo colorido G, um multiconjunto de cores $\mathcal{M}$ (chamado *motivo*), buscamos um subgrafo conexo induzido de G que contém as cores de $\mathcal{M}$. Quando o motivo não pode ser encontrado em G, buscamos uma "ocorrência aproximada" dele, considerando alguns critérios de aproximação. No trabalho mencionado, provou-se que o problema é NP-difícil mesmo que G esteja restrito a árvores e foi proposto um algoritmo exato de enumeração, que enumera apenas motivos pequenos (contendo no máximo 4 vértices). Em 2018, foi proposta uma abordagem baseada em programação inteira para o caso especial em que G está restrito a árvores. No presente trabalho, apresentamos modelos de programação inteira para o referido problema e propomos uma abordagem *branch-and-cut* para o caso geral do problema (quando G é um grafo arbitrário). As restrições de conexidade da solução são adicionadas ao modelo como planos de corte. Com uma pequena adaptação dessa abordagem, obtemos um algoritmo de enumeração. A abordagem apresentada conseguiu resolver instâncias provenientes de redes de interação proteína-proteína contendo, após pré-processamento, aproximadamente 3.000 proteínas (vértices) e 4.100 interações entre elas (arestas).

Palavras-chaves: Busca de motivos. Programação inteira. Motivo em grafos. Branch-and-cut. Redes de interação proteína-proteína.

Abstract

Prazeres, Ricardo Molinari dos. **Integer programming models for the motif search problem in biological networks**. 2022. 81 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2022.

There are several variants of the *motif search* problem in the literature, with many applications in bioinformatics. In the variant called motif search in graphs, proposed in 2006, we are given a colored graph G, a multiset of colors $\mathcal{M}$ (called *motif*) and we seek for a connected induced subgraph of G which contains the colors of $\mathcal{M}$. When the given motif cannot be found in G, we seek for an "approximate match" of it, considering some approximation criteria. In the mentioned work, it was proved that the problem is NP-hard even though G is restricted to trees and an exact enumeration algorithm was proposed, which enumerates only small-size motifs (containing at most 4 vertices). In 2018, an integer programming approach was proposed for the special case where G is restricted to trees. In the present work, we present integer programming models for this problem and propose a *branch-and-cut* approach for the general case of it (when G is an arbitrary graph). The solution connectivity constraints are added to the model as cutting planes. With a small adaptation of this approach, we get an enumeration algorithm. The presented approach was able to solve instances from protein-protein interaction networks containing, after pre-processing, approximately 3,000 proteins (vertices) and 4,100 interactions between them (edges).

Keywords: Motif search. Integer programming. Graph motif. Branch-and-cut. Protein-protein interaction networks.

Lista de figuras

Lista de algoritmos

Lista de tabelas

Lista de abreviaturas e siglas

IPP	interação proteína-proteína
PBMG	problema de busca de motivos em grafos
PEMG	problema de enumeração de motivos em grafos
PI	programação inteira
MPI	modelo de programação inteira
PBM-CM	problema de busca de motivos com cardinalidade mínima
PEMM	problema de enumeração de motivos minimais
SCML	subsequência comum mais longa

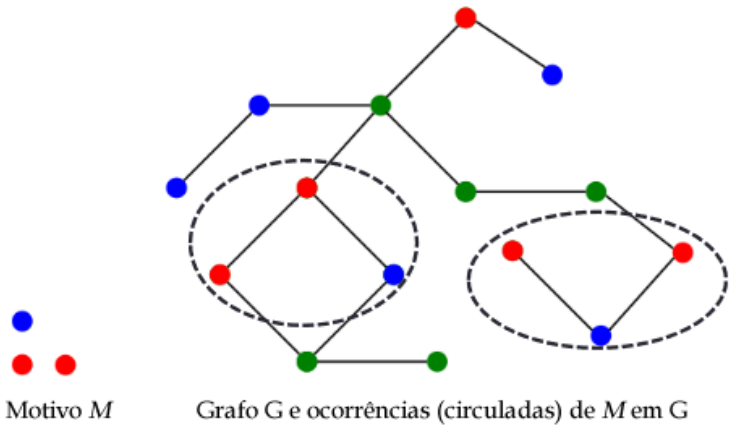
Sumário

1 Introdução

Na área da biologia, *motivos* são padrões que representam aspectos relevantes de dados de DNA e RNA, bem como de redes metabólicas, reguladoras ou de interação proteína-proteína. A identificação, busca e enumeração de motivos dentro de um conjunto de dados biológicos é uma tarefa importante em bioinformática, podendo ajudar a diagnosticar doenças, traçar padrões evolucionários, entre outras aplicações (MALIK; SHARMA, 2014).

Há diferentes definições para um motivo, dependendo da aplicação considerada. Neste trabalho, utilizaremos a definição proposta em Lacroix, Fernandes e Sagot (2006) e, posteriormente, adotada em Lima e Brigatto (2015) e Freire, Lima e Rojas (2018), em que um motivo $\mathcal{M}$ é definido como um multiconjunto de cores, e uma *ocorrência* de $\mathcal{M}$ em uma rede, que é representada por um grafo colorido G, é um subconjunto de vértices R de G, rotulados pelas cores de $\mathcal{M}$, considerando suas multiplicidades, tal que o subgrafo de G induzido por R seja conexo. O problema de busca de motivos em grafos (PBMG) consiste em encontrar uma ocorrência de $\mathcal{M}$ em G, caso exista, e o problema de enumeração de motivos em grafos (PEMG) consiste em enumerar todas as ocorrências de $\mathcal{M}$ em G. Note que o PEMG é uma generalização do PBMG. A Figura 1 ilustra duas ocorrências (circuladas) de um motivo.

Figura 1 – Exemplos de ocorrências de um dado motivo em uma rede biológica representada por um grafo



Fonte: Ricardo Molinari dos Prazeres, 2022

Em Lacroix, Fernandes e Sagot (2006) e Freire, Lima e Rojas (2018), são consideradas aplicações do PEMG/PBMG em redes metabólicas e redes de interação proteína-proteína (redes IPP), respectivamente. Em Lacroix, Fernandes e Sagot (2006), foi provado que mesmo o PBMG em árvores (caso especial em que G é uma árvore) é NP-difícil, e foi proposto um algoritmo para o PEMG. Foram realizados experimentos computacionais utilizando motivos de tamanho igual ou inferior a 4. Em Freire, Lima e Rojas (2018), foi proposto um modelo de programação inteira para o PBMG em árvores e foram apresentados experimentos computacionais que mostraram que o método foi capaz de resolver instâncias reais do problema, com motivos com dezenas de vértices.

Além de buscar ou enumerar ocorrências *exatas* de $\mathcal{M}$ em G, há interesse também nas ocorrências *aproximadas*. Tal interesse é justificado pelo fato de o processo de obtenção dos dados biológicos estar sujeito a falhas, havendo assim um grau de incerteza com relação à existência ou ausência de cada aresta da rede, potencialmente resultando em casos em que $\mathcal{M}$ ocorre de forma exata no organismo em questão, enquanto a rede que o representa contém apenas ocorrências aproximadas de $\mathcal{M}$ em G. Por isso, tanto no PBMG como no PEMG são consideradas também as ocorrências aproximadas. No caso do PBMG, busca-se por uma ocorrência aproximada que se desvie o mínimo possível de uma ocorrência exata, segundo a definição adotada. Quando não há desvio algum, então a ocorrência encontrada é exata, como nas ocorrências ilustradas na Figura 1. Há na literatura diferentes definições de ocorrência aproximada de um motivo em uma rede. Adotamos a definição proposta em Lacroix, Fernandes e Sagot (2006), que é diferente da adotada em Freire, Lima e Rojas (2018), e que será detalhada no Capítulo 3.

1.1 Contribuições

Neste trabalho, apresentamos as seguintes contribuições: propomos três modelos de programação inteira para o PBMG, um para cada caso de representação da rede entrada do problema: caminhos, árvores e grafos arbitrários. Nos três modelos há restrições de conexidade, sendo que no modelo para caminhos adotamos a restrição utilizada em Lima e Brigatto (2015), no modelo de árvores usamos a restrição empregada em Freire, Lima e Rojas (2018) e no modelo para grafos arbitrários adotamos a restrição utilizada em Carvajal *et al.* (2013). Esse último tipo de restrição é a mais complexa entre as três de

implementar, por isso escolhemos o modelo que utiliza grafos arbitrários para empregar a técnica de *branch-and-cut* para solucioná-lo e propomos um algoritmo para resolver o PEMG que, a cada iteração, resolve uma adaptação desse modelo proposto para o PBMG. Apresentamos experimentos computacionais realizados com instâncias geradas a partir de dados reais de redes IPP.

Em relação a trabalhos similares, o método proposto em Lacroix, Fernandes e Sagot (2006) resolve instâncias reais do PEMG, nas quais há motivos com tamanho de no máximo 4 vértices. Já no trabalho de Freire, Lima e Rojas (2018) os motivos chegam a ter dezenas de vértices, porém o modelo proposto por eles só se aplica ao PBMG em árvores, o que faz com que a rede de entrada tenha que ser pré-processada para ser gerada uma nova rede em formato de árvore, podendo vir a comprometer a acurácia do método. Neste trabalho, propomos métodos para solucionar os casos gerais (grafos arbitrários) do PBMG e do PEMG, e os experimentos computacionais mostram que, em ambos os casos, o desempenho do método proposto foi capaz de resolver instâncias em que os motivos podem possuir dezenas de vértices.

Uma vez que no problema de busca de motivo abordado neste trabalho as proteínas são agrupadas por cores, propusemos um método de agrupamento de proteínas no qual descrevemos e implementamos uma adaptação do algoritmo *k-means*.

Submetemos dois artigos referentes ao tema deste trabalho. O primeiro artigo foi submetido no XVIII Simpósio Brasileiro de Sistemas de Informação (SBSI 2022) em 31 de Janeiro de 2022 e conta com 8 páginas em formato de duas colunas. Esse artigo não foi aceito, porque, segundo os revisores, não se adequava à teoria de sistemas de informação, sendo mais indicado à uma conferência de otimização combinatória. A segunda submissão foi para o LIV Simpósio Brasileiro de Pesquisa Operacional (SBPO) em 08 de Abril de 2022 e contém 12 páginas em formato de coluna única. A data de divulgação de trabalhos aceitos dessa conferência é a partir de 25 de Julho de 2022.

1.2 Organização do texto

O restante do texto está organizado da seguinte forma: no Capítulo 2, apresentamos alguns conceitos básicos de teoria dos grafos, redes de fluxo e programação inteira, necessários para a compreensão deste trabalho; no Capítulo 3, definimos formalmente

os problemas abordados neste trabalho e mostramos as principais diferenças entre as definições adotadas aqui e nos trabalhos de Lacroix, Fernandes e Sagot (2006), Lima e Brigatto (2015) e Freire, Lima e Rojas (2018); no Capítulo 4, apresentamos modelos de programação inteira para resolver o PBMG para os casos especiais em que a rede biológica pode ser representada por caminhos, árvores e grafos arbitrários, mostramos como resolver o modelo para grafos arbitrários através do método *branch-and-cut*, e também propomos um algoritmo para resolver o PEMG; no Capítulo 5, mostramos como foram geradas as instâncias a partir de dados reais de redes IPP e que são utilizadas nos experimentos computacionais; no Capítulo 6, apresentamos os resultados dos experimentos computacionais; e, por fim, no Capítulo 7, apresentamos as conclusões e levantamos questões em aberto que podem ser exploradas em trabalhos futuros.

2 Fundamentação teórica

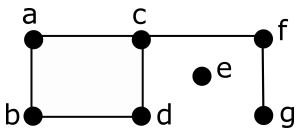
Neste capítulo, apresentamos de forma sintética alguns conceitos básicos de teoria dos grafos, redes de fluxo, programação inteira e o método *branch-and-cut*, utilizado na resolução de um dos modelos de PI propostos neste trabalho.

2.1 Teoria dos grafos

O conteúdo desta seção está baseado no livro de Bondy e Murty (2008) e na apostila de Feofiloff, Kohayakawa e Wakabayashi (2011), que por sua vez, se basearam em Bollobás (1998), Bondy e Murty (1976), entre outras obras.

Um *grafo* G é um par (V, A), sendo V e A conjuntos de elementos denominados *vértices* e *arestas*, respectivamente. Cada aresta uv representa um par não ordenado de vértices distintos. Uma aresta uv *incide* nos vértices u e v, que são chamados de *pontas* dessa aresta. Denotamos por $\bar{A} = \{uv \in V \times V \mid u \neq v \text{ e } uv \notin A\}$ o complemento de A (ou seja, $\bar{A}$ contém os pares não ordenados de vértices para os quais não há uma aresta em A). Por simplicidade, também podemos denotar os vértices e arestas de um grafo qualquer H por $V(H)$ e $A(H)$, respectivamente. A Figura 2 ilustra um grafo com suas arestas e vértices.

Figura 2 – Exemplo de grafo com vértices a, b, c, d, e, f e g e arestas ab, ac, bd, cd, cf e fg

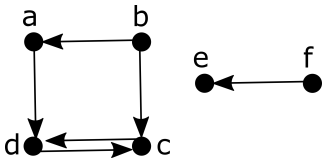


Fonte: Ricardo Molinari dos Prazeres, 2022

Quando as conexões entre os vértices de um grafo representam pares ordenados de vértices distintos, comumente (apesar de não haver consenso) essas são denominadas *arcos*, em vez de arestas. Denominamos o primeiro e segundo vértices de um par ordenado respectivamente por *ponta inicial* e *ponta final* do arco. Denotamos por (u, v) um arco com ponta inicial u e ponta final v. Denominamos os grafos que contêm arcos por grafos *dirigidos* (alguns autores também os denominam por grafos *orientados* ou *digrafos*).

Figura 3 ilustra um grafo orientado com seus arcos e vértices.

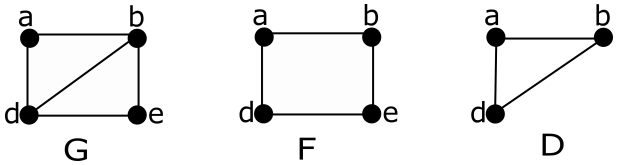
Figura 3 – Exemplo de grafo dirigido com vértices a, b, c, d, e e f e arcos $(b, a), (b, c)$, $(c, d), (d, c)$ e (f, e)



Fonte: Ricardo Molinari dos Prazeres, 2022

Um grafo F é dito um *subgrafo* de G se $V(F) \subseteq V(G)$ e $A(F) \subseteq A(G)$. Dado um subconjunto de vértices $H \subseteq V(G)$, denotamos por $G[H]$ o subgrafo de G *induzido* por H, cujo conjunto de vértices é H e o conjunto de arestas corresponde às arestas de $A(G)$ que ligam dois vértices em H. Na Figura 4, seja $H = \{a, b, d, e\}$, F é subgrafo de G, porém não é induzido pois a aresta bd não está em $A(F)$. Agora, seja $H = \{a, b, d\}$, D é subgrafo induzido de G.

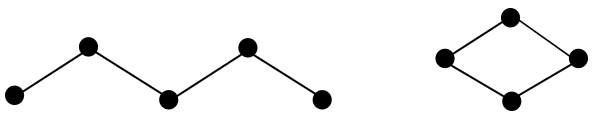
Figura 4 – Exemplo de subgrafo induzido e não induzido



Fonte: Ricardo Molinari dos Prazeres, 2022

Um *caminho* em um grafo G é uma sequência de vértices distintos $C = (v_1, v_2, \cdots, v_n)$, tal que $v_i v_{i+1}$ pertence a $A(G)$ para $i = 1, \cdots, n-1$. Os vértices v_1 e v_n são denominados *extremos* do caminho C. O *comprimento* de um caminho é o número de arestas presentes no mesmo. Um *ciclo* é um grafo O cujo conjunto de vértices admite uma permutação $(v_1, v_2, \cdots, v_n)$ tal que $\{v_1v_2, v_2v_3, \cdots, v_{n-1}v_n\} \cup \{v_nv_1\} = A(O)$, para $n \geq 3$. Um grafo é *acíclico* se não contém nenhum subgrafo que contenha ciclos. A Figura 5 mostra um caminho e um ciclo.

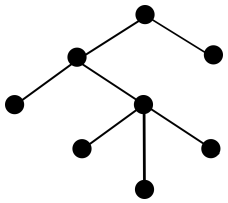
Figura 5 – Exemplos de caminho (esquerda) e ciclo (direita)



Fonte: Feofiloff, Kohayakawa e Wakabayashi (2011)

Um grafo é dito *conexo* se, para qualquer par $\{v, w\}$ de seus vértices, existe um caminho com extremos v e w. Uma árvore é um grafo conexo e acíclico. A Figura 6 ilustra uma árvore.

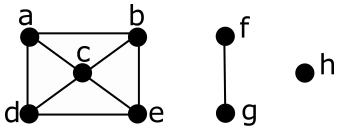
Figura 6 – Exemplo de uma árvore



Fonte: Ricardo Molinari dos Prazeres, 2022

Um subgrafo conexo H de um grafo G é *maximal* se H não é subgrafo próprio de algum subgrafo conexo de G. Uma *componente conexa* (ou por simplicidade apenas componente) de um grafo G é qualquer subgrafo conexo maximal de G. Por consequência, um grafo só é conexo se possui apenas uma componente. A Figura 7 ilustra um grafo com três componentes.

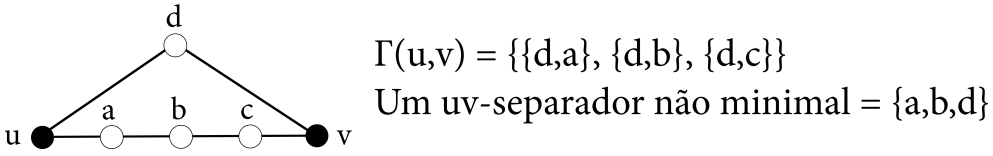
Figura 7 – Um grafo com três componentes conexas



Fonte: Ricardo Molinari dos Prazeres, 2022

Um subconjunto de vértices $S \subset V(G)$ é um *separador* se $G[V \setminus S]$ não for conexo. Dado um par de vértices u e v em V não adjacentes (ou seja, $uv \in \bar{A}(G)$), um conjunto de vértices $S \subseteq V \setminus \{u, v\}$ é um *uv-separador* se não houver caminho entre u e v em $G[V \setminus S]$. Um uv-separador S é *minimal* se não existe nenhum $S' \subset S$, tal que S' é um uv-separador (ver Figura 8). Denotamos por $\Gamma(u, v)$ o conjunto contendo todos os uv-separadores minimais de G.

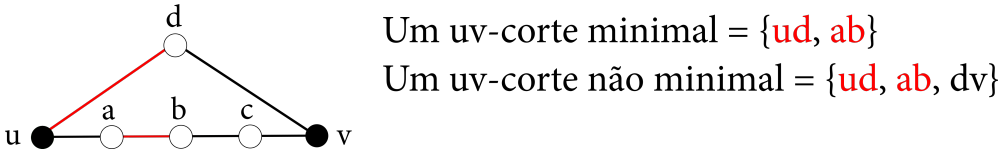
Figura 8 – uv-separadores minimais e não minimal



Fonte: Ricardo Molinari dos Prazeres, 2022

Um subconjunto de arestas $T \subseteq A(G)$ é um *uv-corte* se não há caminho de u a v em $G' = (V, A \setminus T)$. Um uv-corte T é minimal se não existe $T' \subset T$, tal que T' é um uv-corte (ver Figura 9).

Figura 9 – uv-corte minimal e não minimal



Fonte: Ricardo Molinari dos Prazeres, 2022

2.1.1 Coloração e multiconjuntos

Dados um grafo G e um conjunto de cores C, uma *coloração* de G é uma função $f : V(G) \to C$, que atribui a cada vértice de $V(G)$ uma cor de C. Usualmente, em uma coloração, vértices adjacentes devem ter cores diferentes entre si, porém, neste trabalho, a coloração será apenas uma atribuição de cores aos vértices do grafo, não havendo restrições.

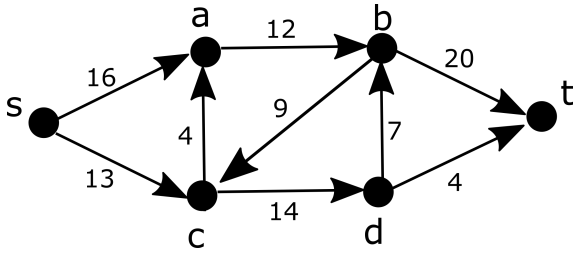
Um multiconjunto é um par (P, m), sendo P um conjunto qualquer e $m : P \to \mathbb{N}_+$ uma função que relaciona a cada elemento p de P um número natural não negativo. A multiplicidade de cada elemento p é dada por $m(p)$.

2.2 Redes de fluxo e fluxos

Os conceitos tratados nesta seção foram obtidos em Cormen *et al.* (2009).

Uma *rede de fluxo* $G = (V, A)$ é um grafo dirigido em que cada arco $(u, v) \in A$ e $u \neq v$ possui capacidade não negativa $c(u, v) \geq 0$ e caso A contenha um arco (u, v), então não pode haver um arco (v, u) (não pode haver arcos antiparalelos). Se $(u, v) \notin A$, definimos $c(u, v) = 0$. Distinguimos os vértices denominados *fonte* s e *sorvedouro* t em uma rede de fluxo e consideramos que cada $u \in V$ está em algum caminho de s a t. Sendo assim, G é conexo. A Figura 10 ilustra uma rede de fluxo. Os valores associados aos arcos indicam a capacidade dos mesmos.

Figura 10 – Exemplo de rede de fluxo com as capacidades dos arcos



Fonte: Adaptado de Cormen *et al.* (2009)

Um fluxo em G é uma função $\varphi : V \times V \to \mathbb{R}$ que satisfaz as duas propriedades a seguir:

$$0 \leq \varphi(u, v) \leq c(u, v), \qquad \forall u, v \in V \tag{1}$$

$$\sum_{v \in V} \varphi(v, u) = \sum_{v \in V} \varphi(u, v), \qquad \forall u \in V \setminus \{s, t\} \tag{2}$$

A propriedade (1) é responsável pela *restrição de capacidade* dos arcos. Ela garante que o fluxo de cada arco seja maior ou igual a zero e não seja maior que sua capacidade. Dizemos que, dado um fluxo $f = \varphi(u, v)$, f é o fluxo que *sai* de u e que *entra* em v, ou apenas o fluxo de u a v . Dito isso, a propriedade (2) é responsável pela *conservação de fluxo* dos arcos, que garante que o fluxo total que entra e sai de cada vértice, excluindo-se s e t, sejam iguais.

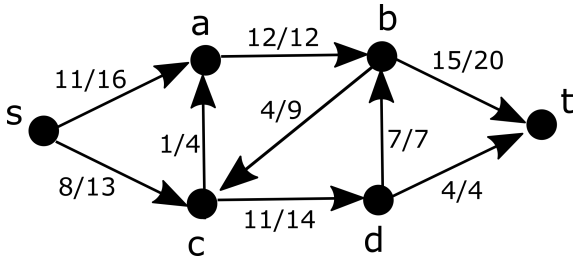
O *valor* $v(\varphi)$ de um fluxo é definido como:

$$v(\varphi) = \sum_{v \in V} \varphi(s, v) - \sum_{v \in V} \varphi(v, s),$$

isto é, o fluxo total que sai da fonte menos o fluxo total que entra na fonte. A Figura 11 ilustra um fluxo φ de valor $v(\varphi) = 19$. À cada arco estão associados $c(u, v)/\varphi(u, v)$. O símbolo "/" é apenas para separar a capacidade e fluxo, não indicando divisão.

No problema denominado *fluxo máximo*, dados uma rede de fluxo, uma fonte e um sorvedouro, o objetivo é encontrar um fluxo de valor máximo.

Figura 11 – Exemplo de um fluxo com as capacidades e fluxos dos arcos



Fonte: Adaptado de Cormen *et al.* (2009)

2.3 Programação inteira

Os conceitos a seguir foram retirados de Conforti, Cornuejols e Zambelli (2014).

Um problema de programação linear inteira, ou apenas Programação Inteira (PI), pode ser definido como: dados dois vetores $c \in \mathbb{R}^m$ e $b \in \mathbb{R}^n$, e uma matriz $A \in \mathbb{R}^{n \times m}$, encontrar um vetor $x \in \mathbb{N}^m$ que maximize (ou minimize) cx, tal que $Ax \leq b$ (ou $Ax \geq b$, no caso de maximização). Segue abaixo a representação de um programa linear inteiro (ou simplesmente *programa inteiro*) de maximização:

$$\begin{aligned} \max \quad & cx \\ s.a \quad & Ax \leq b \\ & x \in \mathbb{N}^m \end{aligned} \tag{3}$$

Note que $Ax \leq b$ é um sistema de inequações (ou *restrições*) lineares. A função linear cx a ser maximizada é chamada de *função objetivo*. Qualquer vetor x que satisfaça as restrições de (3) é uma solução *viável*. Dizemos que (3) é *viável* se existe ao menos uma solução viável. Uma solução x^* viável é *ótima* se $cx^* \geq cx$, para qualquer solução viável x. Diz-se que (3) é uma *formulação* (ou um *modelo*) para um problema de otimização PO se qualquer solução viável x corresponde a uma solução S de PO, sendo que cx deve ser igual ao "ganho" (ou "custo", no caso de minimização) associado à solução S na definição de PO.

Por simplicidade, podemos representar (3) pelo formato a seguir, tal que $S = \{x \in \mathbb{N}^m \mid Ax \leq b\}$:

$$max\{cx \mid x \in S\}.$$

Para obter a *relaxação linear* de (3), substituímos a restrição de integralidade pela restrição de não negatividade, conforme descrito a seguir, sendo que $P_0 = \{x \geq 0 \mid Ax \leq b\}$.

$$\max\{cx \mid x \in P_0\} \tag{4}$$

Em geral, resolver (3) é NP-difícil, enquanto que (4) pode ser resolvido em tempo polinomial. Note que, toda solução viável de (3) é também viável de (4), embora nem toda solução viável de (4) seja também viável de (3). Mais especificamente, somente as soluções fracionárias de (4) não são viáveis de (3). Portanto, se ao resolver (4) encontrarmos uma solução inteira, tal solução será ótima de (3). Caso contrário, tal solução fornecerá um *limitante superior* para (3) (caso seja um problema de minimização, a solução fornecerá um *limitante inferior*).

2.3.1 O método de *branch-and-bound*

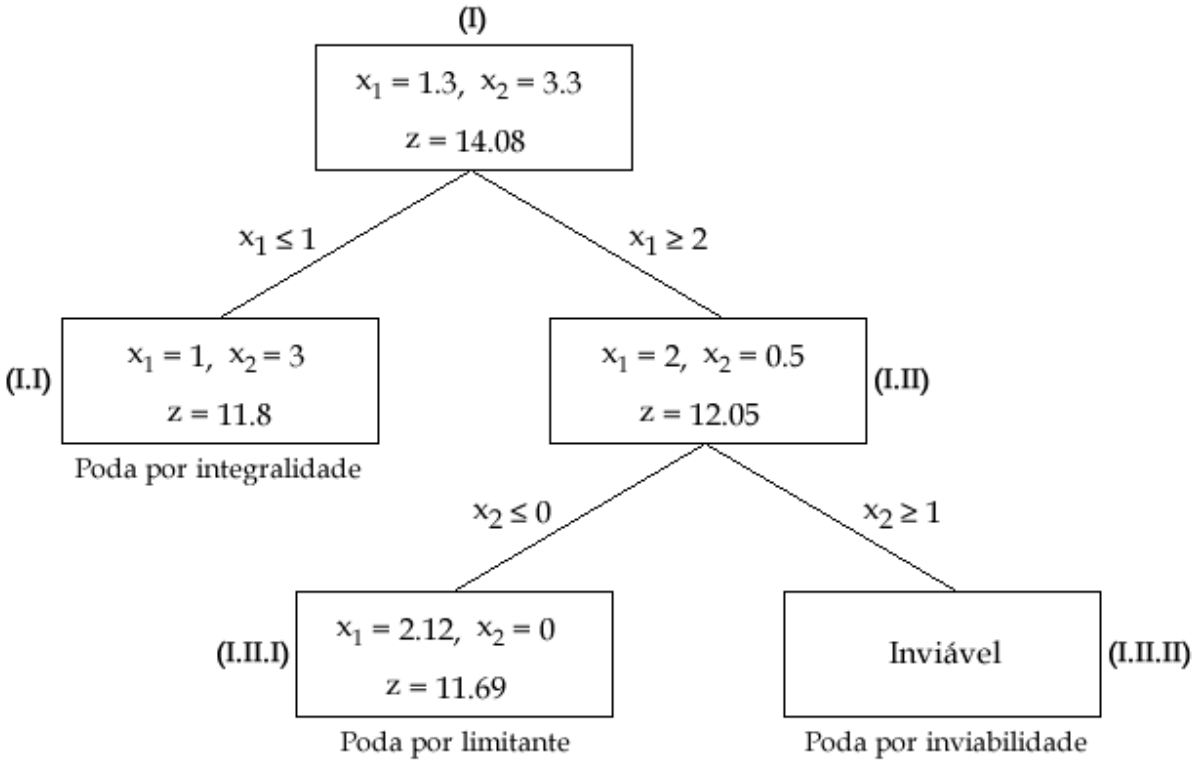
Denominamos por *limitante dual* o valor de uma solução ótima obtido através da relaxação linear de um modelo de programação inteira. Caso a função objetivo desse modelo seja de maximização, o limitante dual é um limitante superior, já se a função objetivo for de minimização, esse é um limitante inferior. Já *limitante primal* é o valor de uma solução viável (inteira) do modelo considerado de PI (relaxado ou não), sendo um limitante inferior se a função objetivo do modelo for de maximização ou superior se essa função for de minimização.

Tipicamente, os resolvedores de PI aplicam o clássico método denominado *branch-and-bound* para resolver (3), utilizando programação linear essencialmente para a obtenção de limitantes (dual e primal).

Utilizamos a Figura 12 para dar uma ideia geral do funcionamento do método de *branch-and-bound*, utilizando o Modelo 5 como exemplo. Depois descreveremos esse método através de um algoritmo.

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ s.a \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x \in \mathbb{N}^m \end{aligned} \tag{5}$$

Figura 12 – Exemplo do método de *branch-and-bound*



Fonte: Adaptado de Conforti, Cornuejols e Zambelli (2014)

Em relação ao funcionamento do *branch-and-bound* ilustrado na Figura 12:

- Encontra-se a solução ótima x^* da relaxação linear de (5), caso x^* seja inteiro, ocorre a denominada *poda por integralidade*, encerrando-se o método com x^* sendo uma solução ótima de (5). Caso contrário, escolhe-se, arbitrariamente ou de acordo com algum critério, uma variável x_j fracionária, sendo que $1 \leq j \leq m$. Conforme representado em (I), a solução da relaxação linear de (5) é $x_1 = 1.3, x_2 = 3.3$ com valor ótimo $z = 14.08$. Portanto, 14.08 é um limitante dual da solução ótima de (5). Após escolher arbitrariamente o valor fracionário de x_1 e realizar a *ramificação* nessa variável, são criados outros dois programas inteiros: (I.I) com a restrição $x_1 \leq 1$ e (I.II) com a restrição $x_1 \geq 2$.
- Em (I.I), a relaxação linear de (5) com restrição adicional $x_1 \leq 1$ tem solução $x_1 = 1, x_2 = 3$ com valor ótimo 11.8. Como essa é uma solução inteira, efetua-se a poda por integralidade e 11.8 é um limitante primal no valor de uma solução ótima de (5) até o momento.
- Já em (I.II), a relaxação linear de (5) com restrição adicional $x_1 \geq 2$ tem solução $x_1 = 2, x_2 = 0.5$ e valor ótimo 12.05. Como esse é o último nó dessa ramificação,

atualizamos o valor do limitante dual global $\bar{z} = 12.05$ e do limitante primal global $\underline{z} = 11.08$. Uma vez que essa solução não é inteira e seu valor é maior que $\underline{z}$, é necessário continuar a busca. Nesse caso, apenas x_2 tem valor fracionário, então, ramificamos nessa variável. Mais dois programas inteiros são gerados, um com a restrição adicional $x_2 \geq 1$, o outro com $x_2 \leq 0$.

- Como em (I.II.II) a relaxação linear é inviável (não há valores para x_1 e x_2 que satisfaçam a solução), nesse caso, ocorre *poda por inviabilidade*.
- Por fim, em (I.II.I), a solução ótima deste programa inteiro relaxado é $x_1 = 2.12$, $x_2 = 0$, com valor ótimo 11.69. Uma vez que esse valor é menor do que $\underline{z}$, este nó sofre *poda por limitante* e o método de *branch-and-bound* refente à (5) está completo, com solução ótima $x_1 = 1, x_2 = 3$ e valor ótimo $z^* = 11.8$.

A seguir, apresentamos o Algoritmo 1, referente ao método de *branch-and-bound*. Considere MPI como modelo de programação inteira.

Algoritmo 1 Método de *branch-and-bound*

procedimento BRANCH-AND-BOUND
  $nós \leftarrow \{nó_0\}$
  $\underline{z} \leftarrow -\infty$
  $x^* \leftarrow \emptyset$
  seja S o conjunto de soluções de MPI
  enquanto $nós \neq \emptyset$ **faça**
    escolha e remova um nó $nó_i$ de $nós$
    $MP_i \leftarrow$ RELAXACAO-LINEAR(MPI_i)
    se RESOLVE(MP_i) não é viável **então**
      continue ▷ poda por inviabilidade
    senão
      $x^i \leftarrow$ SOLUCAO-OTIMA(MP_i)
      $z_i \leftarrow$ VALOR-SOLUCAO(MP_i)
      se $z_i \leq \underline{z}$ **então**
        continue ▷ poda por limitante
      senão
        se $x^i \in \mathbb{N}$ **então** ▷ poda por integralidade
          $x^* \leftarrow x^i$
          $\underline{z} \leftarrow z_i$
        senão ▷ ramificação (*branch*)
          escolha um valor j tal que $x^i_j \notin \mathbb{N}$
          $S_{i_1} := S_i \cap \{x : x_j \leq \lfloor x^i_j \rfloor\}$
          $S_{i_2} := S_i \cap \{x : x_j \geq \lceil x^i_j \rceil\}$
          sejam MPI_{i_1}, MPI_{i_2} e $nó_{i_1}$, $nó_{i_2}$ referentes a S_{i_1} e S_{i_2}
          adicione $nó_{i_1}$ e $nó_{i_2}$ a $nós$
  devolve x^*

Fonte: Adaptado de Conforti, Cornuejols e Zambelli (2014)

2.3.2 O método de planos de corte

Considere S e P_0 os conjuntos de soluções de (3) e (4), respectivamente. Definimos x^0 como uma solução ótima de (4) e z_0 o valor dessa solução. Uma inequação $\alpha u \leq \beta$ é *válida* para um conjunto $K \subseteq \mathbb{R}^d$ se for satisfeita por todo ponto $\bar{u} \in K$. Uma inequação válida $\alpha x \leq \beta$ para S que é violada por x^0 é um *plano de corte* (ou por simplicidade apenas *corte*) que separa x^0 de S. Seja $\alpha x \leq \beta$ um plano de corte e defina

$$P_1 := P_0 \cap \{x : \alpha x \leq \beta\}.$$

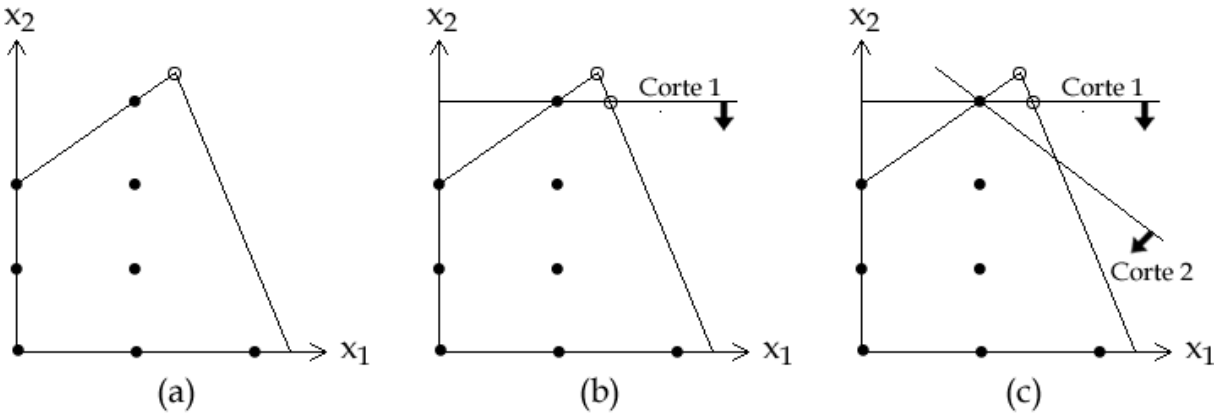
Uma vez que $S \subseteq P_1 \subset P_0$, a relaxação de PI baseada em P_1 é mais forte que a relaxação baseada em P_0, no sentido de que o valor ótimo do programa linear

$$max\{cx : x \in P_1\}$$

é um limitante superior no valor z^* pelo menos tão bom quanto z_0, já que a solução ótima x^0 não pertence a P_1. O *método de planos de corte* consiste na aplicação iterativa dessa abordagem.

A Figura 13 ilustra a aplicação de dois cortes em um programa inteiro, cujo conjunto de soluções é representado pelos pontos com preenchimento. Em (a), uma solução do referente programa inteiro relaxado é representada pelo ponto sem preenchimento. Em (b), essa solução é eliminada pelo *Corte 1*, mantendo-se todos as soluções inteiras da solução. Esse procedimento é repetido através do *Corte 2* em (c).

Figura 13 – Aplicação de cortes



Fonte: Adaptado de Conforti, Cornuejols e Zambelli (2014)

O método de planos de corte é utilizado no caso em que a matriz A de (3) possui um número imenso de linhas, o que tornaria impraticável a resolução mesmo de sua relaxação linear. Isso ocorre quando o número de restrições do modelo de PI em questão é exponencial no tamanho da entrada do problema.

Considere o programa linear abaixo:

$$\begin{aligned} \min \quad & cx \\ s.a \quad & A'x \geq b' \\ & Dx \geq d \\ & x \geq 0 \end{aligned} \tag{6}$$

Note que (6) e (4) são equivalentes, porém em (6) estamos considerando dois conjuntos distintos de restrições, sendo que assume-se que a quantidade de restrições descritas por $Dx \geq d$ é "imensa". Considere PL' o programa linear gerado deixando-se de incluir uma ou mais restrições de $Dx \geq d$. Abaixo apresentamos o Algoritmo (2), referente

ao método de planos de corte. A finalidade do procedimento ADICIONAR-MAIS-CORTES será explicado em seguida.

Algoritmo 2 Método de planos de corte

1: **procedimento** MÉTODO-PLANOS-CORTE
2: $i \leftarrow 1$
3: $M_i \leftarrow PL'$
4: **repita**
5: seja x_i^* uma solução ótima de M_i
6: **se** $Dx_i^* \geq d$ **então**
7: **devolve** x_i^* ▷ x_i^* é uma solução ótima de (6)
8: **senão**
9: seja $\alpha x \geq \beta$ uma inequação de $Dx^* \geq d$, tal que $\alpha x_i^* < \beta$
10: $M_{i+1} \leftarrow M_i \cap \alpha x \geq \beta$
11: $i \leftarrow i + 1$
12: **enquanto** ADICIONAR-MAIS-CORTES = VERDADEIRO

Fonte: Ricardo Molinari dos Prazeres, 2022

O chamado *problema da separação* consiste em determinar se existe alguma restrição em $Dx \geq d$ violada por x^* e, em caso afirmativo, encontrar uma delas (isso é feito na linha 9 do Algoritmo (2). Grötschel, Lovász e Schrijver demonstraram em Grötschel, László e Schrijver (1981) que quando o problema da separação pode ser resolvido em tempo polinomial, então o método de planos de corte para resolver (6) termina em tempo polinomial. Portanto, é possível resolver programas lineares de forma eficiente, mesmo que eles tenham uma quantidade exponencial de restrições, desde que o problema da separação possa ser resolvido de forma eficiente.

O método de planos de corte pode ser executado enquanto houver cortes para serem adicionados, porém em alguns casos, seguindo determinados critérios, pode ser desejável encerrar o método antes de adicionar todos os cortes ou de verificar se ainda há mais cortes. Essa verificação é realizada pelo procedimento ADICIONAR-MAIS-CORTES executado na linha 12 do Algoritmo (2). Também pode ser preferível que ao invés de adicionar um único corte, como na linha 10 do Algoritmo (2), gerar vários cortes e adicionar todos em M_i para criar M_{i+1}. Podemos adicionar cortes em diferentes momentos durante a resolução de um programa inteiro, nesta seção mostraremos os cortes sendo aplicados durante a execução do método denominado *branch-and-cut*.

2.3.3 O método de *branch-and-cut*

O método *branch-and-cut* funciona de forma semelhante ao método *branch-and-bound* para programas inteiros, porém com uma etapa de planos de corte antes do passo de ramificação no *branch-and-bound*.

A seguir, apresentamos o Algoritmo 3, referente ao método de *branch-and-cut*.

Algoritmo 3 Método de *branch-and-cut*

1: **procedimento** BRANCH-AND-CUT
2: $nós \leftarrow \{nó_0\}$
3: $\underline{z} \leftarrow -\infty$
4: $x^* \leftarrow \emptyset$
5: seja S o conjunto de soluções de MPI
6: **enquanto** $nós \neq \emptyset$ **faça**
7: escolha e remova um nó $nó_i$ de $nós$
8: $MP_i \leftarrow$ RELAXACAO-LINEAR(MPI_i)
9: **se** RESOLVE(MP_i) não é viável **então**
10: **continue**
11: **senão**
12: $x^i \leftarrow$ SOLUCAO-OTIMA(MP_i)
13: $z_i \leftarrow$ VALOR-SOLUCAO(MP_i)
14: **se** $z_i \leq \underline{z}$ **então**
15: **continue**
16: **senão**
17: **se** $x^i \in \mathbb{N}$ **então**
18: $x^* \leftarrow x^i$
19: $\underline{z} \leftarrow z_i$
20: **senão** ▷ verifica se adiciona cortes antes de ramificar
21: **se** ADICIONA-MAIS-CORTES=VERDADEIRO **então**
22: ADICIONA-CORTES
23: vá para linha 9
24: escolha um valor j tal que $x^i_j \notin \mathbb{N}$
25: $S_{i_1} := S_i \cap \{x : x_j \leq \lfloor x^i_j \rfloor\}$
26: $S_{i_2} := S_i \cap \{x : x_j \geq \lceil x^i_j \rceil\}$
27: sejam MPI_{i_1}, MPI_{i_2} e $nó_{i_1}$, $nó_{i_2}$ referentes a S_{i_1} e S_{i_2}
28: adicione $nó_{i_1}$ e $nó_{i_2}$ a $nós$
29: **devolve** x^*

Fonte: Adaptado de Conforti, Cornuejols e Zambelli (2014)

3 Descrição do problema

Neste capítulo, realizamos uma revisão na literatura referente a problemas relacionados à busca de motivos em redes biológicas, definimos formalmente os problemas de busca e enumeração de motivos propostos neste trabalho e comparamos nossas definições com as dos trabalhos mais semelhantes ao nosso encontrados durante a revisão.

3.1 Problema de busca de motivos em redes biológicas

Há, na literatura, diversas aplicações para o problema de busca de motivo em redes biológicas. Nesta seção, inicialmente apresentamos um artigo que faz uma revisão sistemática referente a esse problema, categorizando as principais abordagens, técnicas, vantagens, desvantagens, entre outras características. Em seguida, é realizada uma pesquisa focada em trabalhos que utilizam técnicas de PI aplicadas ao problema.

O artigo de Hashim, Mabrouk e Al-Atabany (2019) é uma revisão sistemática que lista 119 diferentes algoritmos de busca de motivos em DNA, dividindo-os em quatro categorias: enumerativos, probabilísticos, combinatórios e baseados em computação natural (conceitos e aplicações da computação inspirados na natureza); e em diversas subcategorias. De acordo com o texto, de modo geral, algoritmos de *enumeração* consistem em um conceito simples de busca exaustiva, com garantia de encontrar todos os motivos, no entanto, são lentos e requerem muitos parâmetros, o que aumenta a dificuldade de serem aplicados em redes e/ou motivos grandes. Já as abordagens *probabilísticas* são mais rápidas que as enumerativas, lidando melhor com redes e/ou motivos grandes, porém não realizam buscas globais na rede e, portanto, não conseguem encontrar todos os motivos. Técnicas baseadas em computação natural podem ser rápidas, lidar bem com alto número de dados e realizar buscas globais, contudo com elas não há garantia de que todos os motivos serão localizados na rede. Por fim, a abordagem combinatória utiliza dois ou mais algoritmos, podendo esses serem de categorias diferentes (probabilístico e combinatório, por exemplo). A Tabela 1 foi adaptada do trabalho de Hashim, Mabrouk e Al-Atabany (2019) e mostra uma comparação entre as categorias descritas (pontos de interrogação indicam que pode ser sim ou não). As colunas “Enum.”, “Prob.”, “Natural” e “Comb” significam respectivamente: enumerativos, probabilísticos, baseados em computação natural e combinatórios. A coluna “Grande nº de dados” indica se o algoritmo lida bem com grande número de dados.

Tabela 1 – Comparação entre as categorias de algoritmos de busca de motivos

	Enum.	Prob.	Natural	Comb.
Técnica	Exaustivo	Heurística	Meta-heurística	Mais de uma
Grande nº de dados	Não	Sim	Não	?
É rápido	Não	Sim	?	?
Encontra ótimo global	Sim	Não	Não	?
Encontra todos os motivos	Sim	Não	?	?

Fonte: Hashim, Mabrouk e Al-Atabany (2019)

3.1.1 Algoritmos que utilizam técnicas de PI

Apesar de ser um artigo recente, Hashim, Mabrouk e Al-Atabany (2019) não faz nenhuma referência a algoritmos que utilizam técnicas de PI. Os artigos de Sandve e Drablos (2006) e Makolo (2016) também realizam revisões sistemáticas quanto à busca de motivos em redes biológicas, porém neles também não há referências a algoritmos de PI. Portanto, foi adicionada neste trabalho uma revisão específica de trabalhos que utilizam PI no problema de busca de motivos.

A Tabela 2 indica algumas características que se destacaram nos trabalhos encontrados. A sigla *IPP* corresponde a interação proteína-proteína. As siglas da coluna "Tipo de busca" serão explicadas a seguir, quando é dado um panorama geral sobre os trabalhos que a utilizam. A coluna "Topologia" indica se o trabalho utiliza a topologia do motivo como um dos parâmetros da busca.

Descreveremos inicialmente os trabalhos que, apesar de utilizarem PI, abordam problemas de busca de motivo que destoam do problema tratado neste texto.

- Os trabalhos de Dinh, Rajasekaran e Kundeti (2011), Yu *et al.* (2012), Kingsford, Zaslavsky e Singh (2011), Tang *et al.* (2011) e Zaslavsky e Singh (2006) abordam o problema denominado (l, d)-busca de motivo ((l, d)-BM), que consiste em: dado um número n de sequências biológicas e dois valores inteiros l e d, encontrar todas as *strings* M de comprimento l tais que M ocorre em cada uma das n sequências, com no máximo d incompatibilidades. Nesse contexto, considera-se M o motivo. De modo geral, nesses trabalhos, a utilização de PI é aplicada ao encontrar o alinhamento de sequência múltipla local, sem lacunas e de comprimento fixo que minimiza uma medida de distância de soma-de-pares.

Tabela 2 – Comparação entre trabalhos que utilizam PI em busca de motivos

Tipo de busca	Rede biológica	Topologia	Referência
(l, d)-BM	DNA e Proteína	Sim	(ZASLAVSKY; SINGH, 2006)
(l, d)-BM	DNA	Sim	(TANG *et al.*, 2011)
(l, d)-BM	DNA	Sim	(KINGSFORD; ZASLAVSKY; SINGH, 2011)
(l, d)-BM	DNA	Sim	(YU *et al.*, 2012)
(l, d)-BM	DNA, RNA e Proteína	Sim	(DINH; RAJASEKARAN; KUNDETI, 2011)
SCPM	IPP	Sim	(DITTRICH *et al.*, 2008)
SCPM	IPP e Metabólica	Sim	(BACKES *et al.*, 2011)
DCP	IPP	Sim	(MAZZA *et al.*, 2016)
PE	Proteína	Sim	(HATTORI *et al.*, 2019)
PE	RNA	Sim	(POOLSAP; KATO; AKUTSU, 2009)
PE	RNA	Sim	(LEGENDRE; ANGEL; TAHI, 2018)
PBM	IPP	Não	(BRUCKNER *et al.*, 2010)
PBM-MinCC	IPP	Não	(FREIRE; LIMA; ROJAS, 2018)
Max-Motif	-	Não	(LIMA; BRIGATTO, 2015)

Fonte: Ricardo Molinari dos Prazeres, 2021

- Backes *et al.* (2011) e Dittrich *et al.* (2008) aplicaram um problema denominado subgrafo conexo de peso máximo (SCPM) à busca de motivo, que consiste em: dado um grafo dirigido G com pesos em cada um de seus vértices, deseja-se encontrar subgrafos conexos de G de tamanho k que maximizem a soma dos pesos dos vértices.
- Mazza *et al.* (2016) tratam de um problema denominado detecção de complexos proteicos (DCP), no qual, dada uma rede de interação proteína-proteína representada por um grafo G e um subgrafo relacionado a doenças H de G, almeja-se localizar conjuntos de proteínas com alto índice de interação em H. A utilização de PI é dada ao buscar as pontuações mais altas (maximizar) segundo o índice de interação.
- Poolsap, Kato e Akutsu (2009) e Legendre, Angel e Tahi (2018) realizam predição da estrutura (PE) secundária de RNAs a partir de subestruturas dos mesmos, já Hattori *et al.* (2019) estuda a PE da estrutura tridimensional de proteínas, tendo como base suas próprias estruturas primárias. Apesar de utilizarem técnicas de PI em redes biológicas, esses estudos não são focados em encontrar ocorrências de um dado padrão em redes biológicas.
- Bruckner *et al.* (2010) descreveram algumas variantes para o PBMG proposto por Lacroix, Fernandes e Sagot (2006) e já descrito neste trabalho. A utilização de PI é feita em apenas uma delas, a qual permite um número n de inserções de vértices ao motivo buscado com intuito de flexibilizar a busca de ocorrência. Nessa variante, dada uma rede representada por um grafo G com peso em suas arestas e um motivo

$\mathcal{M}$ de tamanho l, o objetivo é encontrar uma ocorrência conexa H de tamanho $l + n$ de $\mathcal{M}$ em G, tal que o peso total das arestas de H seja máximo. Pelo fato desse objetivo não ter semelhança com o da variante abordada por nós, não enfatizaremos esse trabalho como faremos com os próximos.

A seguir descreveremos de forma sucinta os trabalhos que, além de utilizarem PI, abordam variantes do problema de busca de motivo com objetivos semelhantes ao do nosso trabalho. Esses trabalhos serão melhor abordados na Seção 3.3.

- Na variante denominada Max-Motif para o PBMG, proposta por Lima e Brigatto (2015), o objetivo é encontrar uma ocorrência conexa H de um dado motivo $\mathcal{M}$ cujo multiconjunto de cores H mais se "aproxima" de $\mathcal{M}$. Para isso, a função objetivo do modelo de PI proposta maximiza a quantidade de vértices selecionados de H, mas é adicionado ao modelo uma restrição que impede que a quantidade de vértices selecionados de uma determinada cor não ultrapasse a quantidade que esta cor aparece em $\mathcal{M}$. Nesse trabalho, os autores não definiram um tipo específico da rede biológica.
- Freire, Lima e Rojas (2018) propuseram uma variante para o PBMG, denominada problema de busca de motivos com número mínimo de componentes conexas (PBM-MinCC), na qual, em vez da saída do problema ser um grafo conexo, passa-se a ser um grafo com número mínimo de componentes, cujos vértices correspondem às cores do motivo. Essa variante é exclusiva para o caso especial em que a rede de entrada é uma árvore e será melhor explanada adiante.

Outro aspecto dos trabalhos de Lima e Brigatto (2015), Bruckner *et al.* (2010) e Freire, Lima e Rojas (2018) é que são os únicos, dentre os trabalhos que utilizam PI, que não consideram a topologia do motivo buscado como critério da busca. Bruckner *et al.* (2010) justificam essa escolha baseados no fato de que, na prática, muitas vezes essa informação não é de fato conhecida, razão pela qual também não consideramos a topologia da rede como critério na busca do motivo.

3.1.2 Categorização dos problemas

Com base na revisão apresentada acima, concluímos ser interessante dividir os problemas abordados em duas categorias principais:

(1) - problemas que denominaremos por *descoberta* de motivos (DINH; RAJASEKARAN; KUNDETI, 2011), (YU *et al.*, 2012), Kingsford, Zaslavsky e Singh (2011), (TANG *et al.*, 2011), (ZASLAVSKY; SINGH, 2006), (BACKES *et al.*, 2011), (DITTRICH *et al.*, 2008) e (MAZZA *et al.*, 2016): nesses trabalhos o objetivo é identificar novos motivos, ou seja, inicialmente não se sabe quais são motivos que serão localizados na rede biológica. Os métodos de descoberta geralmente consistem em encontrar n ocorrências de um dado tamanho de um subgrafo $\mathcal{M}$ (possível motivo) em um determinado grafo e verificar se n é substancialmente maior que o número de ocorrências de $\mathcal{M}$ em um grafo aleatório (CIRIELLO; GUERRA, 2008).

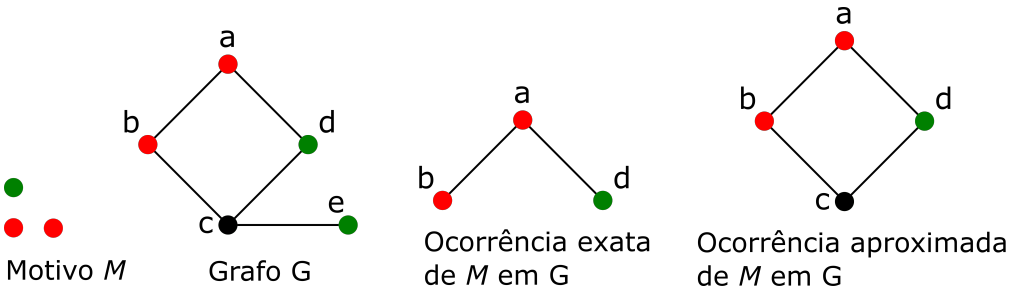
(2) - problemas que denominaremos por *busca* de motivos (LIMA; BRIGATTO, 2015), (BRUCKNER *et al.*, 2010) e (FREIRE; LIMA; ROJAS, 2018): já nesses trabalhos, o objetivo é encontrar uma ocorrência exata ou aproximada (a definição de ocorrência aproximada muda entre os trabalhos) de um dado motivo em uma rede. Como já mencionado, esses três trabalhos são variantes do PBMC, proposto por Lacroix, Fernandes e Sagot (2006), que, de modo simplificado, consiste em: dado um grafo colorido G, encontre todos o subgrafos (ocorrências) conexos de G cujo conjunto de vértices contenha as cores (de modo exato ou aproximado) de um dado multiconjunto de cores (motivo). Em Lacroix, Fernandes e Sagot (2006) também foi provado que esse problema é NP-completo mesmo para encontrar uma única ocorrência do motivo, com uma rede de entrada restrita a árvores e com a restrição adicional de que não pode haver repetição de cores no motivo. Os problemas de busca e enumeração de motivos propostos nesta dissertação são variações dos problemas de busca de motivos.

3.2 Definição do problema

Seja C um conjunto de cores e seja $G = (V, A, \xi)$ um grafo colorido conexo, em que V é o conjunto de vértices, A é o conjunto de arestas e $\xi : V \to C$ é uma função (ou *coloração*) que atribui uma cor $\xi(u) \in C$ para cada vértice $u \in V$.

Um *motivo* é um multiconjunto de cores $\mathcal{M} = (C, m)$, onde $m : C \to \mathbb{N}$ é a função de multiplicidade das cores de C em $\mathcal{M}$. Ou seja, $m(c)$ indica a quantidade de elementos de $\mathcal{M}$ com a cor c, para todo $c \in C$. Seja $H \subseteq V$ um subconjunto de vértices, denotamos por $n(H, c)$ o número de vértices em H com a cor c. Dizemos que H é uma *ocorrência* de $\mathcal{M}$ em G se $G[H]$ é conexo e $n(H, c) \geq m(c)$, para toda cor $c \in C$. Adicionalmente, se $n(H, c') > m(c')$ para pelo menos uma cor $c' \in C$, dizemos que H é uma ocorrência *aproximada*. Caso contrário (ou seja, $n(H, c) = m(c), \forall c \in C$), dizemos que H é uma ocorrência *exata* (ver Figura 14).

Figura 14 – Ocorrência exata e aproximada de um motivo em um grafo colorido.



Fonte: Ricardo Molinari dos Prazeres, 2022

Se H for uma ocorrência aproximada de $\mathcal{M}$ em G, dizemos que H possui $|H| - |\mathcal{M}|$ vértices *extras*. A seguir, enunciamos o problema de busca de motivos no qual estamos interessados:

Problema 1 (Busca de Motivos de Cardinalidade Mínima)**.** *Dado um grafo colorido conexo $G = (V, A, \xi)$ e um motivo $\mathcal{M} = (C, m)$, encontrar um subconjunto $H \subseteq V$, tal que $n(H, c) \geq m(c)$, para todo $c \in C$, $G[H]$ seja conexo e $|H|$ seja mínimo.*

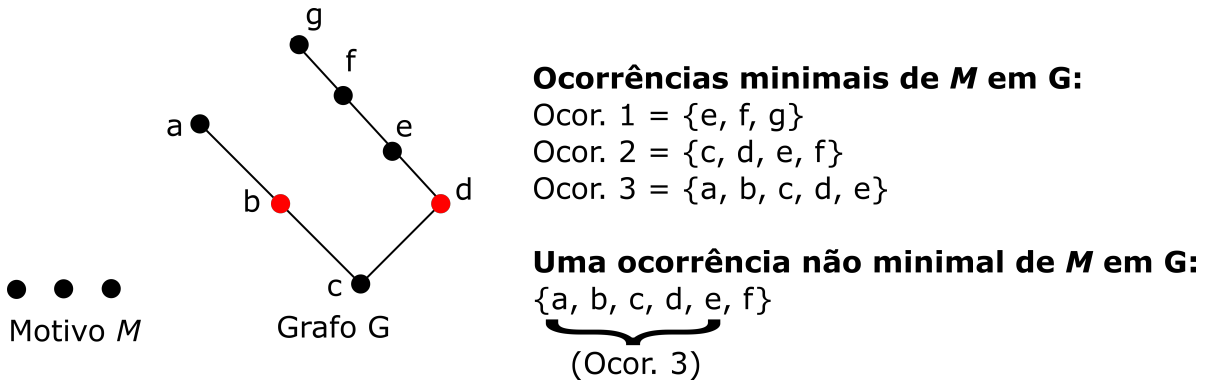
Por simplicidade, assumimos que $n(V, c) \geq m(c)$, para todo $c \in C$ (consequentemente, vale que $|V| \geq |\mathcal{M}|$). Note que, quando essa condição não é satisfeita, não há ocorrência de $\mathcal{M}$ em G. No Problema de Busca de Motivos de Cardinalidade Mínima (PBM-CM), busca-se por uma ocorrência H de $\mathcal{M}$ em G que "se desvie" o mínimo possível de uma ocorrência exata. Quando $|H| = |\mathcal{M}|$, então H é uma ocorrência exata. Caso contrário (ou seja, quando $|H| > |\mathcal{M}|$), H é uma ocorrência aproximada com número mínimo de vértices extras.

Uma ocorrência H de $\mathcal{M}$ em G é *minimal* se não existe nenhuma ocorrência H' de $\mathcal{M}$ em G tal que $H' \subseteq H$ (ver Figura 15). Note que nem toda ocorrência minimal tem

cardinalidade mínima, mas toda ocorrência de cardinalidade mínima é minimal. A seguir, enunciamos o Problema de Enumeração de Motivos Minimais (PEMM):

Problema 2 (Enumeração de Motivos Minimais)**.** *Dado um grafo colorido conexo* $G = (V, A, \xi)$ *e um motivo* $\mathcal{M} = (C, m)$*, listar todas as ocorrências minimais de* $\mathcal{M}$ *em* G*.*

Figura 15 – Ocorrências minimais e não-minimal de um motivo.



Fonte: Ricardo Molinari dos Prazeres, 2022

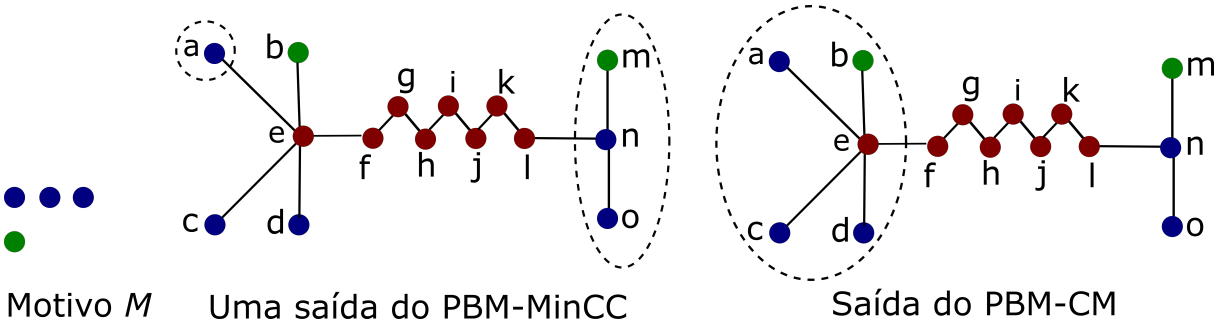
3.3 Comparação com trabalhos semelhantes

Nas seções a seguir são realizadas comparações entre o PBM-CM e PEMM com os problemas apresentados em Freire, Lima e Rojas (2018), Lima e Brigatto (2015) e Lacroix, Fernandes e Sagot (2006).

3.3.1 PBM-MinCC

Em Freire, Lima e Rojas (2018), é proposta uma outra definição em que exige-se que, em uma ocorrência aproximada H de $\mathcal{M}$ em G, deve valer que $|H| = |\mathcal{M}|$, porém $G[H]$ pode ser desconexo. Os autores propõem então o *Problema de Busca de Motivos com número Mínimo de Componentes Conexas* (PBM-MinCC), em que busca-se por uma ocorrência H de $\mathcal{M}$ em G, tal que o número de componentes conexas de $G[H]$ seja mínimo (ver Figura 16). Note que, nas duas definições, se G contiver ao menos uma ocorrência exata H^* de $\mathcal{M}$, então H^* será uma solução ótima.

Figura 16 – Soluções ótimas para o PBM-CM e PBM-MinCC, considerando a mesma entrada



Fonte: Ricardo Molinari dos Prazeres, 2022

Nas aplicações do problema de busca ou enumeração de motivos na área de biologia, o interesse em ocorrências aproximadas é justificado pelo fato de o processo de obtenção dos dados biológicos estar sujeito a falhas, havendo assim um grau de incerteza com relação à existência ou ausência de cada aresta da rede, potencialmente resultando em casos em que $\mathcal{M}$ ocorre de forma exata no organismo em questão, enquanto a rede que o representa contém apenas ocorrências aproximadas de $\mathcal{M}$ em G. Pode ocorrer, por exemplo, de um par de vértices (u, v) representar duas proteínas que, no organismo em estudo, possuem relação entre si, porém por alguma falha no processo de geração dos dados, a aresta uv não estar incluída em A.

Sejam H_1 e H_2 soluções ótimas do PBM-MinCC e PBM-CM, respectivamente, para uma mesma entrada $(G, \mathcal{M})$. No PBM-MinCC, minimiza-se o número de componentes de $G[H_1]$. Dessa forma, garante-se que $G[H_1]$ possui o número mínimo de "arestas faltantes". Porém, como as componentes de $G[H_1]$ podem estar muito afastadas, considerando a estrutura de G, a probabilidade de essas componentes estarem conectadas no organismo em estudo pode ser muito pequena.

Já no PBM-CM, minimiza-se a cardinalidade de H_2 e exige-se que $G[H_2]$ seja conexo. Note que se H_2 for uma ocorrência aproximada, haverá $|H_2| - |\mathcal{M}|$ vértices extras, que estão em H_2 apenas para garantir a conexidade de $G[H_2]$. Se tais vértices forem removidos, o subgrafo resultante pode vir a ter um número de componentes maior do que $G[H_1]$, como ocorre no exemplo mostrado na Figura 16. Porém, como o número de vértices extras é mínimo, garante-se que tais componentes estejam próximas, considerando a estrutura de G, o que pode aumentar a probabilidade de essas componentes estarem conectadas no

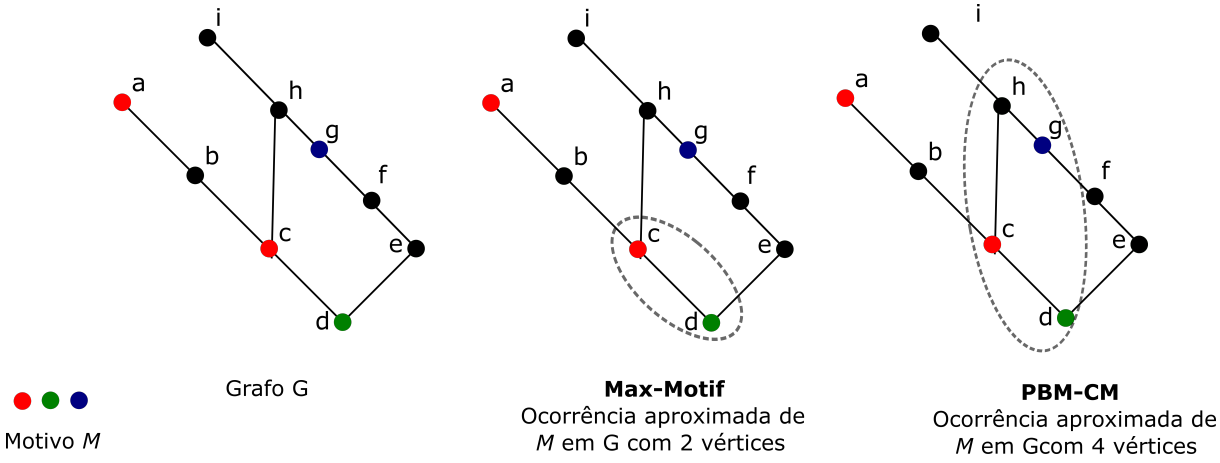
organismo em estudo. Não encontramos na literatura nenhum estudo empírico comparando essas duas abordagens.

3.3.2 Max-Motif

Em Lima e Brigatto (2015), é proposto um problema de busca de motivos em que a definição de uma ocorrência L de um motivo $\mathcal{M}$ é semelhante à deste trabalho, com a diferença de que L deve conter o mesmo número ou menos vértices que $\mathcal{M}$, ou seja $|L| \leq |\mathcal{M}|$ (no nosso trabalho, uma ocorrência de $\mathcal{M}$ deve ter cardinalidade maior ou igual à de $\mathcal{M}$). Os autores propuseram um modelo de PI, no qual a função objetivo é maximizar o número de vértices de L com uma restrição que impede que esse número seja maior que $|\mathcal{M}|$.

Note que essa função tem objetivo oposto à do nosso modelo, em que queremos justamente minimizar o número de vértices de uma ocorrência H de $\mathcal{M}$. Porém em ambos os modelos, caso haja uma ocorrência com o mesmo número de vértices que $\mathcal{M}$ (ocorrência exata), significa que $|H| = |L|$ e essa é uma solução ótima para os dois problemas (Max-Motif e PBM-CM). A Figura 17 ilustra saídas para esses problemas, dada uma mesma instância.

Figura 17 – Soluções "semelhantes" para o PBM-CM e o problema de Max-Motif, considerando a mesma instância

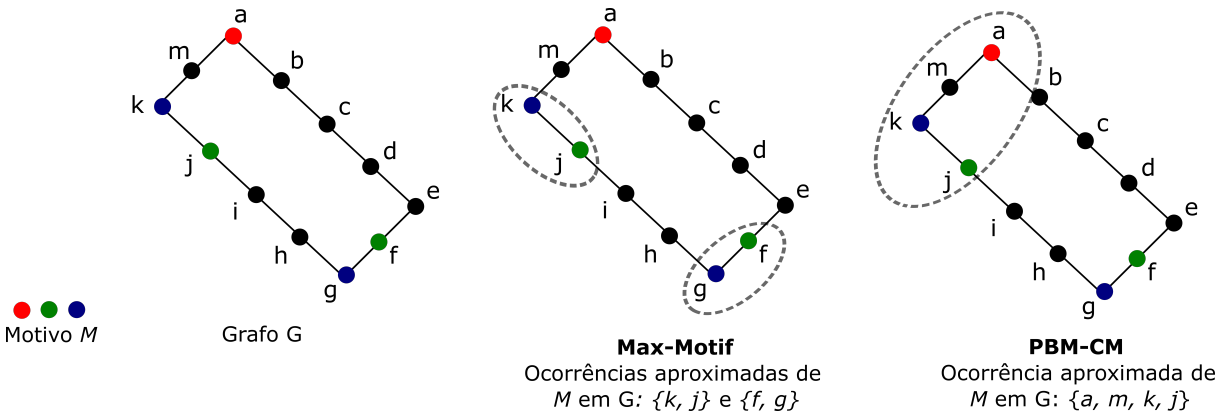


Fonte: Ricardo Molinari dos Prazeres, 2022

Na Figura 17, os autores não utilizam a definição de *ocorrência aproximada* para o problema de Max-Motif, mas nós a utilizamos com o significado de ocorrência com número

de vértices menor que o do motivo buscado. Note que nas ocorrências encontradas para os dois problemas, bastaria que houvesse no organismo em estudo (mesmo não havendo em G por uma possível falha) uma ligação entre o vértice g e c ou entre g e d para que o motivo $\mathcal{M}$ estivesse contido de modo exato nele. Contudo, mostramos a seguir através da Figura 18 exemplos em que as saídas para os problemas podem ser menos semelhantes entre si.

Figura 18 – Outros exemplos de soluções para o PBM-CM e para o problema de Max-Motif, considerando a mesma instância



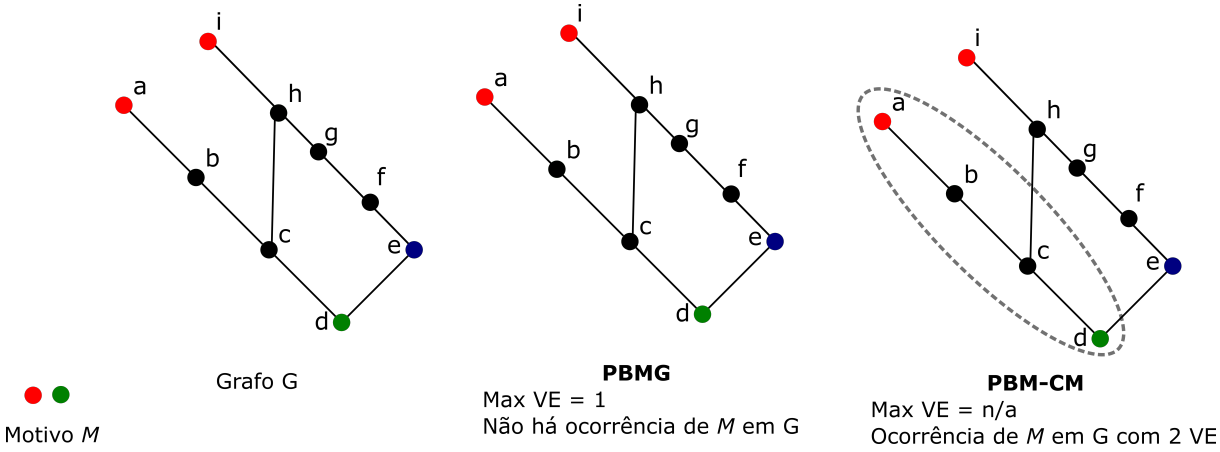
Fonte: Ricardo Molinari dos Prazeres, 2022

Na Figura 18, note que no caso das ocorrências $\{k, j\}$ e $\{a, m, k, j\}$ há a mesma situação do exemplo da Figura 17, sendo necessário apenas uma ligação entre os vértices a e k ou a e j para existência de uma ocorrência exata. Porém, no caso da saída $\{g, f\}$, há uma maior distância entre esses vértices e a, podendo haver menos chance de existir ligação entre eles no organismo em questão.

3.3.3 PBMG

Por fim, em relação ao PBMG, proposto em Lacroix, Fernandes e Sagot (2006) e já descrito neste trabalho, é utilizado o número máximo de vértices extras (Max VE na Figura 19) que uma solução pode conter como um parâmetro do problema. A Figura 19 ilustra saídas para o PBM-CM e para o PBMG considerando a mesma instância.

Figura 19 – Soluções para o PBM-CM e PBMG, considerando a mesma instância



Fonte: Ricardo Molinari dos Prazeres, 2022

Na Figura 19, no PBMG, quando a quantidade máxima estipulada de vértices extras é 1, não há ocorrências do motivo buscado. Já no PBM-CM, esse parâmetro não se aplica e sempre é encontrada uma solução, caso o motivo esteja contido de modo conexo na rede.

4 Modelos de programação inteira

Neste capítulo, apresentamos os modelos de PI propostos para resolver o PBMG, mostramos como resolver um desses modelos através do método *branch-and-cut*. Em seguida, propomos um algoritmo para o PEMM que, a cada iteração, resolve uma adaptação desse modelo.

4.1 Modelos de PI para o PBM-CM

Consideramos três modelos distintos de PI para o PBM-CM, que são detalhadamente explicados a seguir, cada um para um caso de rede de entrada do problema: quando essa pode ser representada por caminhos, árvores e grafos arbitrários. Basicamente, a diferença entre esses modelos está na forma como é realizada a restrição de conexidade da solução do problema.

A seguir estão a função objetivo e as restrições que são comuns para os três modelos propostos. Denotamos por $V_c = \{v \in V \mid \xi(v) = c\}$ o conjunto dos vértices de G com cor c. Seja x_u uma variável binária, para todo vértice $u \in V$, com a interpretação de que $x_u = 1$ se e somente se o vértice u foi selecionado para compor a solução ótima.

$$\min \sum_{v \in V} x_v \tag{7}$$

$$s.a \sum_{v \in V_c} x_v \geq m(c), \qquad \forall c \in C \tag{8}$$

$$x_v \in \{0, 1\}, \qquad \forall v \in V \tag{9}$$

A função objetivo (7) minimiza o número de vértices selecionados para compor a solução, fazendo com que ela tenha cardinalidade mínima.

A restrição (8) garante que a multiplicidade com que cada cor ocorre na solução seja maior ou igual à no motivo buscado.

A restrição (9) é responsável pela integralidade da variável.

4.1.1 Caso em que o grafo de entrada é um caminho

Considere G o grafo que representa a rede de entrada do problema. No caso especial em que G é um caminho, utilizamos a restrição de conexidade adotada no trabalho de Lima e Wakabayashi (2011), no qual tem a função de garantir uma coloração convexa.

$$x_p - x_q + x_r = 1, \qquad \forall p, q, r \in \{1, \ldots, n\} \mid p < q < r \tag{10}$$

De modo informal, pode-se entender a restrição (10) da seguinte maneira: caso os vértices p e r, sendo $r \geq p + 2$, forem selecionados para compor a solução, então todos os vértices q, para $q = p + 1, \ldots, r - 1$, também deverão ser selecionados. Considere n como o índice de maior valor dentre os índices dos vértices selecionados para compor a solução.

Denotamos por $\mathrm{F}_{\text{PBMC-CM}}$ a formulação composta por (7), (8), (9) e (10).

4.1.2 Caso em que o grafo de entrada é uma árvore

No caso especial em que o grafo de entrada G do problema é uma árvore, utilizamos a restrição de conexidade utilizada em Freire, Lima e Rojas (2018), que por sua vez, basearam-se no modelo proposto em Chopra *et al.* (2017), para o problema de recoloração convexa de árvores. Nas restrições apresentadas a seguir, utilizamos uma variável binária y_{uv} para toda aresta $y_{uv} \in A$, indicando as arestas selecionadas para a solução.

$$\sum_{v \in V} x_v - \sum_{uv \in A} y_{uv} = 1 \tag{11}$$

$$y_{uv} \leq x_u, \qquad \forall uv \in A \tag{12}$$

$$y_{uv} \leq x_v, \qquad \forall uv \in A \tag{13}$$

$$y_{uv} \in \{0, 1\}, \qquad \forall uv \in A \tag{14}$$

A restrição (11) faz com que a diferença entre o número de vértices e arestas selecionados, que corresponde à quantidade de componentes da solução, seja igual a 1, garantindo que a solução seja um subgrafo conexo, baseado nos seguintes argumentos: sabendo-se que nesse caso particular, o grafo de entrada é uma árvore, pode-se assumir que cada uma de suas componentes c_i, para $i = 1, 2, ..., k$ possui uma quantidade de arestas $|ca_i|$ igual a seu número de vértices $|cv_i| - 1$:

$$|ca_i| = |cv_i| - 1,$$

portanto:

$$\sum_{i=1}^{k}(|cv_i| - 1) = \sum_{i=1}^{k}|cv_i| - k.$$

As restrições (12) e (13) são responsáveis pela vinculação das variáveis x e y, de modo que cada variável y_{uv} só possa ser igual a 1 se e somente se $x_u = x_v = 1$ (uma aresta só pode ser selecionada caso suas pontas também sejam selecionadas).

A restrição (14) é responsável pela integralidade da variável.

Denotamos por $\mathrm{F}_{\text{PBMA-CM}}$ a formulação composta por (7), (8), (9), (11), (12), (13) e (14).

4.1.3 Caso em que o grafo de entrada é arbitrário

Por fim, apresentamos a seguir a restrição de conexidade para o caso em que o grafo que representa a rede é arbitrário.

$$\sum_{w \in S} x_w \geq x_u + x_v - 1, \qquad \forall uv \in \bar{A}, \forall S \in \Gamma(u, v) \tag{15}$$

Considere o subconjunto $H^* = \{u \in V \mid x_u^* = 1\}$ como uma solução ótima do PBM-CM para a instância dada $(G, \mathcal{M})$. Basicamente, essa restrição garante que para todo par de vértices u e v em H^*, seja escolhido pelo menos um vértice de cada uv-separador, garantindo que em $G[H^*]$ haja caminho entre todos pares u e v, e por consequência que $G[H^*]$ seja conexo.

Essa restrição foi proposta em Carvajal *et al.* (2013), que utilizaram modelos de PI para solucionar um problema de planejamento florestal, que visa a selecionar uma região de floresta conectada, de forma a maximizar o número de espécies e habitats protegidos. Essa restrição está baseada no seguinte lema:

Lema 1. *Dado um subconjunto de vértices $H \subseteq V$, o subgrafo $G[H]$ é conexo se e somente se, para todo par de vértices u e v em H, vale que $S \cap H \neq \emptyset$, para todo uv-separador $S \in \Gamma(u, v)$.*

Denotamos por $\mathrm{F}_{\text{PBMG-CM}}$ a formulação composta por (7), (8), (9) e (15).

Note que a quantidade de separadores em G pode ser exponencial no tamanho de G, sendo assim impraticável incluir todas as inequações do tipo (15) a priori no modelo. Por isso, aplicaremos o método *branch-and-cut* para resolver $\mathrm{F}_{\text{PBMG-CM}}$.

4.1.4 Visão dos três modelos

Na Figura 20 estão as três formulações de PI para o PBM-CM propostas neste capítulo. A função objetivo e as restrições apresentadas antes da linha pontilhada são comuns para os três modelos. As restrições apresentadas após a linha pontilhada são referentes ao requisito de conexidade da solução.

Figura 20 – Formulações de PI para o PBM-CM para cada estrutura que representa a rede de entrada: caminho, árvore e grafo arbitrário

$\mathrm{F}_{\text{PBMC-CM}}$	$\mathrm{F}_{\text{PBMA-CM}}$	$\mathrm{F}_{\text{PBMG-CM}}$
$\min \sum_{v \in V} x_v$	$\min \sum_{v \in V} x_v$	$\min \sum_{v \in V} x_v$
$s.a \sum_{v \in V_c} x_v \geq m(c),\ \forall c \in C$	$s.a \sum_{v \in V_c} x_v \geq m(c),\ \forall c \in C$	$s.a \sum_{v \in V_c} x_v \geq m(c),\ \forall c \in C$
$x_v \in \{0,1\},\ \forall v \in V$	$x_v \in \{0,1\},\ \forall v \in V$	$x_v \in \{0,1\},\ \forall v \in V$
$x_p - x_q + x_r = 1,$ $1 \leq p < q < r \leq n$	$\sum_{v \in V} x_v - \sum_{uv \in A} y_{uv} = 1$	$\sum_{w \in S} x_w \geq x_u + x_v - 1,$ $\forall uv \in \bar{A}, \forall S \in \Gamma(u,v)$
	$y_{uv} \leq x_u, \quad \forall uv \in A$	
	$y_{uv} \leq x_v, \quad \forall uv \in A$	
	$y_{uv} \in \{0,1\}, \quad \forall uv \in A$	

Fonte: Ricardo Molinari dos Prazeres, 2022

4.2 Aplicando o método branch-and-cut

Conforme visto na Seção 2.3.2, uma das etapas do método *branch-and-cut* consiste em resolver a relaxação linear de $\mathrm{F}_{\text{PBMG-CM}}$, denotada por RL, em que a restrição de integralidade (9) é substituída pela restrição $0 \leq x_u \leq 1$, para todo $u \in V$. Além disso,

ao invés de serem incluídas a priori, as inequações (15) são incluídas como planos de corte. Ou seja, inicialmente, considera-se o modelo RL_0, que corresponde a RL sem as inequações do tipo (15). A cada iteração i, temos que resolver o problema da separação, que consiste em, dada uma solução ótima x^* de RL_i, encontrar uma inequação do tipo (15) violada por x^*, caso exista, e incluí-la no modelo, de forma que RL_{i+1} corresponda RL_i acrescido da inequação encontrada. O procedimento termina quando x^* satisfizer todas as inequações do tipo (15), ainda que nem todas inequações desse tipo tenham sido incluídas explicitamente no modelo.

Conforme visto na Seção 2.3.2, se o problema da separação puder ser resolvido em tempo polinomial, então esse procedimento terminará em tempo polinomial. A seguir, mostramos como resolver o problema da separação das inequações (15) em tempo polinomial.

Dado um subconjunto de vértices $S \subseteq V$, denotamos por $x^*(S) = \sum_{u \in S} x^*_u$ a soma dos valores das variáveis associadas aos vértices de S na solução x^*. Observe que o problema da separação pode ser enunciado como um problema de otimização, conforme a seguir.

Problema 3 (Separação)**.** *Dada uma solução ótima x^* de RL_i, encontrar um par de vértices $uv \in \bar{A}$ e um uv-separador $S \in \Gamma(u, v)$, tais que $z(u, v, x^*, S) = x^*(S) - x^*_u - x^*_v$ seja mínimo.*

Seja (u, v, S) uma solução ótima do Problema 3. Note que se $z(u, v, x^*, S) < -1$, então a inequação $x^*(S) \geq x^*_u + x^*_v - 1$ está violada por x^*. Caso contrário (ou seja, se $z(u, v, x^*, S) \geq -1$), como $z(u, v, x^*, S)$ é mínimo, vale que $z(u', v', x^*, S') \geq -1$, para todo $u'v' \in \bar{A}$ e $S' \in \Gamma(u', v')$, o que implica que x^* satisfaz todas as inequações do tipo (15), mesmo que parte delas não tenha sido adicionada a RL_i.

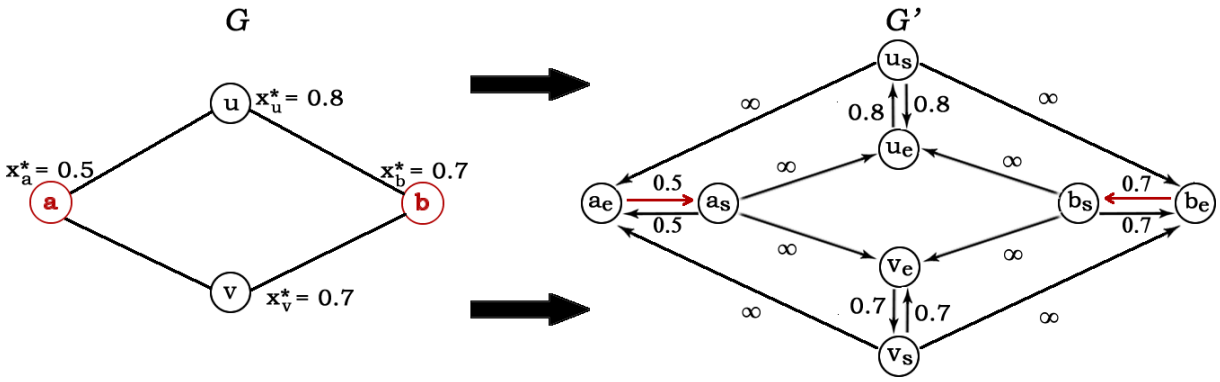
A maneira proposta em Carvajal *et al.* (2013) para resolver o Problema 3 consiste em, a partir de G e x^*, construir um grafo dirigido $G' = (V', A')$ com capacidade nos arcos e resolver o problema de encontrar um uv-corte mínimo para diversos pares $u, v \in V'$. A seguir, mostramos como G' é construído e, em seguida, apresentamos um algoritmo para resolver o problema da separação.

Para cada vértice $u \in V$, são adicionados dois vértices u_e e u_s em V'. Dizemos que u_e é um *vértice de entrada* e u_s é um *vértice de saída*. Além disso, são adicionados a A' dois arcos (u_e, u_s) e (u_s, u_e), ambos com capacidade igual a x^*_u. Para cada aresta

$uv \in A$, são adicionados a A' dois arcos (u_s, v_e) e (v_s, u_e), ambos com capacidade igual a ∞. A capacidade de um arco (u, v) é denotada por $c(u, v)$ e a soma das capacidades de um conjunto de arcos T é denotado por $c(T) = \sum_{(u,v)\in T} c(u, v)$.

Note que, para qualquer uv-separador S em G, existe um $u_s v_e$-corte T correspondente em G', tal que para cada vértice $w \in S$ existe um arco $(w_e, w_s) \in T$. Além disso, vale que $x^*(S) = c(T)$. Por exemplo, na Figura 21, o subconjunto de vértices $S = \{a, b\}$ de V é um uv-separador em G, tal que $x^*(S) = 0.5 + 0.7 = 1.2$, e o subconjunto de arcos $T = \{(a_e, a_s), (b_e, b_s)\}$ de A' é o $u_s v_e$-corte correspondente em G', tal que $c(T) = 0.5 + 0.7 = 1.2$.

Figura 21 – Construção de G' a partir de G e x^*.



Fonte: Ricardo Molinari dos Prazeres, 2022

O Algoritmo 4, apresentado a seguir, resolve o Problema 3.

Algoritmo 4 Resolve o problema da separação

```
1:  procedimento RESOLVE-SEPARACAO
2:      G' ← CONSTRÓI-GRAFO-DIRIGIDO(G, x*)
3:      para cada uv ∈ Ā faça
4:          se x*_u + x*_v > 1 então
5:              T ← CORTE-MÍNIMO(G', u_s, v_e)
6:              se c(T) − x*_u − x*_v < −1 então
7:                  S ← {u ∈ V | (u_e, u_s) ∈ T}
8:                  adicione a restrição Σ_{w∈S} x_w ≥ x_u + x_v − 1
9:                  devolve verdadeiro
10:     devolve falso
```

Fonte: Ricardo Molinari dos Prazeres, 2022

Assumimos que a sub-rotina CONSTRÓI-GRAFO-DIRIGIDO constrói o grafo G', conforme explicado acima, e a rotina CORTE-MÍNIMO devolve um $u_s v_e$-corte mínimo

em G'. Há diversos algoritmos de tempo polinomial para resolver o problema de fluxo máximo-corte mínimo (CORMEN *et al.*, 2009).

Note que, se a condição na linha 3 não for satisfeita (ou seja, $x^*_u + x^*_v \leq 1$), então para qualquer $u_s v_e$-corte T em G', como $c(T) \geq 0$, vale que $c(T) - x^*_u - x^*_u \geq -1$. Portanto, nesse caso, não há necessidade de encontrar um $u_s v_e$-corte mínimo em G'. Em nossa implementação, o conjunto $\bar{A}$ é varrido em ordem decrescente, de forma que uv é listado antes de $u'v'$ se e somente se $x^*_u + x^*_v \geq x^*_{u'} + x^*_{v'}$. Assim, se a condição da linha 4 não for satisfeita, o algoritmo pode ser interrompido.

Se alguma inequação violada por x^* for encontrada, a mesma é adicionada a RL_{i+1} na linha 8. Caso contrário, na linha 10 o algoritmo devolve *falso*, indicando que x^* satisfaz todas as inequações do tipo (15) e, portanto, não há mais necessidade de incluir planos de corte.

Em nossa implementação, optamos por utilizar o algoritmo de Ford-Fulkerson (CORMEN *et al.*, 2009) na função Corte-mínimo, por ser um algoritmo de fácil implementação e ter demonstrado bom desempenho nas instâncias consideradas.

Durante o procedimento de *branch-and-cut*, além de resolver RL para obter um limitante inferior no valor ótimo, os resolvedores também encontram, através de heurísticas, soluções inteiras viáveis de RL_i, a fim de obter um limitante superior no valor ótimo. Isso pode ocorrer numa etapa em que $i < j$, onde j é a última iteração do método de planos de corte aplicado na resolução de RL. Como consequência, as soluções inteiras encontradas dessa forma podem violar a restrição de conexidade, o que comprometeria a corretude do procedimento de *branch-and-cut*.

Para evitar que esse tipo de erro venha a acontecer, temos que adicionar uma *lazy constraint* toda vez que isso acontecer, o que pode ser feito resolvendo-se o problema da separação para uma solução x' inteira viável de RL_i. Nesse caso específico, é possível resolver o Problema 3 de forma mais eficiente do que o algoritmo apresentado acima. Em nossa implementação, fazemos isso através de uma simples adaptação do algoritmo de busca em profundidade (CORMEN *et al.*, 2009), cuja descrição será omitida, por simplicidade de exposição.

Também é importante salientar que caso o grafo G que representa a rede de entrada não seja conexo, basta executar o algoritmo de *branch-and-cut* em cada componente conexa de G.

4.3 Um algoritmo para resolver o PEMM

A seguir, propomos um algoritmo para resolver o PEMM que, através de uma adaptação no modelo $\mathrm{F}_{\text{PBMG-CM}}$, a cada iteração encontra uma nova ocorrência H de $\mathcal{M}$ em G, tal que H é minimal. Embora não seja uma exigência na definição de PEMM, as ocorrências são listadas em ordem crescente de cardinalidade.

Seja $\mathcal{H}$ um conjunto de ocorrências de $\mathcal{M}$ em G. Denotamos por $\mathrm{F}_{\text{PEMM}}(\mathcal{H})$ o modelo $\mathrm{F}_{\text{PBMG-CM}}$ acrescido da seguinte restrição:

$$\sum_{v \in H} x_v \leq |H| - 1, \forall H \in \mathcal{H} \tag{16}$$

Se $\mathrm{F}_{\text{PEMM}}(\mathcal{H})$ for viável, dada uma solução ótima x^* de $\mathrm{F}_{\text{PEMM}}(\mathcal{H})$, a restrição (16) garante que o subconjunto $H^* = \{u \in V \mid x^*_u = 1\}$ é uma ocorrência de $\mathcal{M}$ em G, tal que $H^* \not\supseteq H$, para todo $H \in \mathcal{H}$. Como $|H^*|$ é mínimo, vale que H^* é minimal. Dessa forma, $\mathrm{F}_{\text{PEMM}}(\mathcal{H})$ é viável se e somente se existe alguma ocorrência minimal de $\mathcal{M}$ em G que não está contida em $\mathcal{H}$. Além disso, note que $|H^*| \leq |H|$, para toda ocorrência minimal H de $\mathcal{M}$ em G que não está contida em $\mathcal{H}$.

A seguir, apresentamos o Algoritmo 5, que enumera todas as ocorrências minimais de $\mathcal{M}$ em G em ordem crescente de cardinalidade.

Algoritmo 5 Enumera ocorrências minimais

```
1: procedimento RESOLVE-PEMM
2:     H ← ∅
3:     enquanto F_PEMM(H) é viável faça
4:         x* ← uma solução ótima de F_PEMM(H)
5:         H ← {u ∈ V | x*_u = 1}
6:         H ← H ∪ {H}
7:     devolve H
```

Fonte: Ricardo Molinari dos Prazeres, 2022

5 Geração de instâncias a partir de dados biológicos

Neste capítulo, descrevemos como as instâncias dos problemas PBM-CM e PEMM, utilizadas neste trabalho foram obtidas a partir de dados reais de redes IPP. Além disso, descrevemos o algoritmo que utilizamos para agrupar proteínas e como medimos a qualidade dos agrupamentos gerados.

Nos dados biológicos brutos obtidos em ⟨http://igm.univ-mlv.fr/AlgoB/gramofone⟩, temos redes IPP e motivos a serem buscados ou enumerados nessas redes. Cada proteína da rede é representada por um vértice em V e as interações entre elas são representadas pelas arestas em A. Os motivos são simplesmente conjuntos de proteínas.

A cada aresta $uv \in A$ está associado um valor p_{uv} no intervalo $(0, 1]$, indicando a probabilidade de as proteínas representadas por u e v interagirem de fato no organismo representado pela rede G. Quanto menor o valor de p_{uv}, menor a probabilidade de as proteínas representadas por u e v interagem. Optamos por considerar apenas as arestas com probabilidade pelo menos 0.5. Em consequência disso, alguns vértices podem passar a ter grau zero (ou seja, nenhuma aresta incidindo nos mesmos). Caso isso ocorra, tais vértices são removidos de V. Além disso, as proteínas representadas pelos vértices removidos que aparecem nos motivos a serem buscados ou enumerados na rede são removidas dos mesmos.

Para realizar as tarefas de busca e enumeração da forma como definidas neste trabalho, precisamos atribuir uma cor a cada proteína da rede e dos motivos. Basicamente, definimos uma função para medir a similaridade entre as proteínas presentes na rede e nos motivos, posteriormente aplicamos um algoritmo de agrupamento dessas proteínas, sendo que as proteínas de um mesmo grupo serão representadas com a mesma cor na coloração ξ de G e no multiconjunto de cores $\mathcal{M}$.

5.1 Similaridade entre proteínas

Segundo Shyu e Tsai (2009), o nível de identidade funcional ou estrutural de duas proteínas pode ser medido através do alinhamento das sequências de aminoácidos que descrevem tais proteínas. Um método clássico utilizado para realizar esta tarefa é encontrar uma *Subsequência Comum Mais Longa* (SCML) de tais sequências, havendo na literatura

diversos algoritmos eficientes para resolver esse problema (BERGROTH; HAKONEN; RAITA, 2000).

Dada uma sequência de caracteres $X[1..m]$, uma *subsequência* $S[1..s]$ de $X[1..m]$ é obtida excluindo-se $m - s$ caracteres de X. Dadas duas sequências de caracteres X e Y, dizemos que S é uma *subsequência comum* de X e Y se S é uma subsequência tanto de X como de Y. Se S tiver comprimento máximo (ou seja, se $|S| \geq |S'|$, para toda subsequência comum S' de X e Y), dizemos que é uma SCML de X e Y. Denotamos por $scml(X, Y)$ o comprimento de uma SCML de X e Y. Considere o exemplo abaixo:

$X = [A, \mathbf{C}, \mathbf{D}, B, \mathbf{A}, \mathbf{C}]$, $Y = [B, \mathbf{C}, \ \mathbf{D}, \mathbf{A}, A, \mathbf{C}, D]$
Subsequências comuns de X e Y: $[B, C], [A, C], [D, A, C], \ldots$
Uma SCML de X e Y: $S = [C, D, A, C]$, com $scml(X, Y) = |S| = 4$

A função de similaridade ϕ atribui um valor real indicando a similaridade entre duas sequências X e Y da seguinte forma:

$$\phi(X, Y) = \frac{1}{2} \times \frac{scml(X, Y)}{|X| + |Y|}$$

5.2 Agrupamento de proteínas

5.2.1 Algoritmo baseado no *k-means*

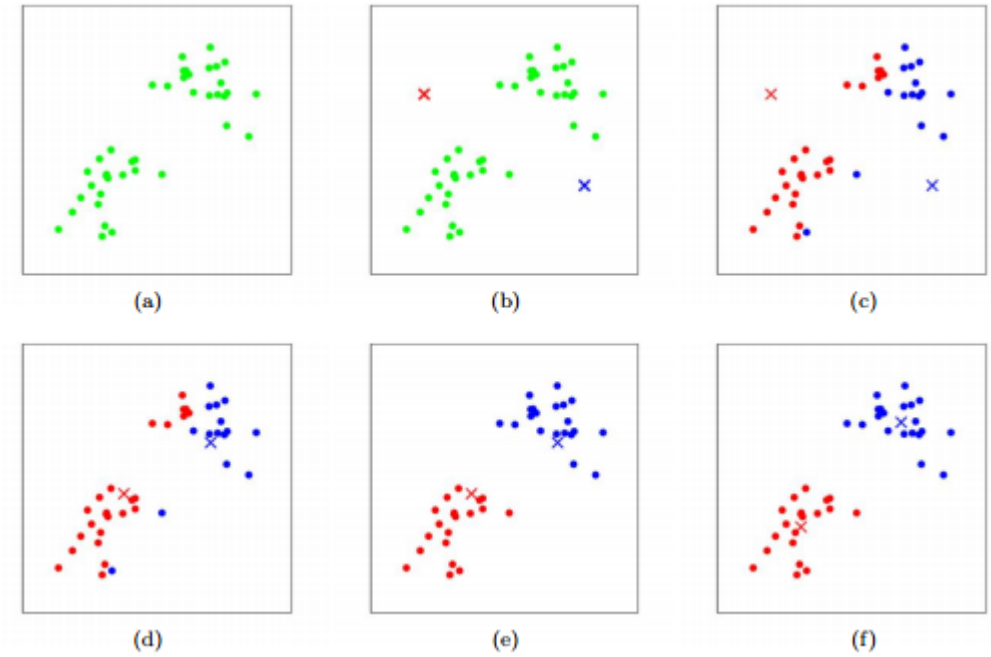
O *k-means* é um dos métodos de agrupamento mais utilizados em aplicações científicas (EDLA *et al.*, 2018). Nele, cada grupo é representado por seu centroide, que é obtido calculando a posição média entre os objetos do grupo. De modo simplificado (mais detalhes são apresentados adiante), seu algoritmo consiste em: (I) selecione aleatoriamente k pontos iniciais como centroides; (II) gere k grupos, atribuindo cada objeto do conjunto de dados ao centroide mais próximo a ele; (III) recalcule o centroide de cada grupo; (IV) caso não haja alteração dos centroides, retorne os grupos, caso contrário, volte ao passo (II). A Figura 22 ilustra esse processo, no qual (a) representa o conjunto de dados inicial, (b) a escolha arbitrária de $k = 2$ centroides e (c-f) as iterações do algoritmo até sua convergência.

Nós utilizamos a similaridade entre cada par de proteínas como critério de "distância" entre elas. Como no *k-means* quanto menor a distância entre dois objetos é maior a chance

deles estarem em um mesmo grupo, utilizamos o valor inversamente proporcional à similaridade como distância, ou seja, a distância entre duas proteínas p_1 e p_2 $dist(p1, p2)$ é:

$$dist(p1, p2) = \frac{1}{\phi(p_1, p_2)}$$

Figura 22 – Inicialização e iterações do *k-means*



Fonte: Yang, Towey e Zhou (2019)

Pelo fato de que essa distância não foi obtida do modo convencional, as propriedades geométricas que dão embasamento para a escolha do centroide como ponto médio podem não existir. Portanto, optamos por não gerar centroides da maneira tradicional do *k-means*, de modo que os centroides só podem ser representados pelos próprios objetos do grupo. O Algoritmo 6 realiza o agrupamento de proteínas, considerando essa adaptação em relação ao *k-means*. Assumimos que a sub-rotina GERA-CENTROIDES-ALEATORIOS gera k centroides aleatórios a partir do conjunto de dados.

Note que em 6, caso o algoritmo não convirja no número máximo de iterações estipulado, é retornado falso, indicando que isso ocorreu. Como veremos adiante, esse algoritmo é executado diversas vezes, então caso não haja convergência, apenas descartamos tal execução.

Algoritmo 6 Realiza agrupamento de proteínas

Entrada: um conjunto de vértices $V\{v_1, \ldots, v_n\}$; uma função $dist(v_1, v_2)$ que retorna a distância entre dois vértices; o número de grupos k que será devolvido e o número máximo de iterações $maxIter$

Saída: um agrupamento dos vértices de V

```
1: procedimento REALIZA-AGRUPAMENTO
2:     centroides ← GERA-CENTROIDES-ALEATORIOS(V, k)
3:     l ← 0
4:     enquanto l < maxIter faça
5:         para cada v_i ∈ V, v_i ∉ centroides faça
6:             minDist ← ∞
7:             centroideMinDist ← −1
8:             para cada v_c ∈ centroides faça
9:                 se dist(v_i, v_c) < minDist então
10:                    minDist ← dist(v_i, v_c)
11:                    centroideMinDist ← v_c
12:            grupos[centroideMinDist].adiciona(v_i)
13:        para cada grupo ∈ grupos faça
14:            minTotalDistProt ← ∞
15:            novoCentroide ← −1
16:            para cada v_i ∈ grupos faça
17:                totalDistProt ← 0
18:                para cada v_j ≠ v_i ∈ grupos faça
19:                    totalDistProt ← totalDistProt + dist(v_i, v_j)
20:                se totalDistProt < minTotalDistProt então
21:                    minTotalDistProt ← totalDistProt
22:                    novoCentroide ← v_i
23:            novosCentroides.adicona(novoCentroide)
24:        se centroides = novosCentroides então
25:            devolve grupos
26:        l ← l + 1
27:        centroides ← novosCentroides
28:    devolve falso
```

Fonte: Ricardo Molinari dos Prazeres, 2022

5.2.2 Pontuação do agrupamento

No método *k-means*, a escolha do número de grupos que serão formados e as posições dos centroides iniciais são muito importantes. Como geralmente a escolha dos valores desses parâmetros levam a resultados de agrupamento diferentes, uma única execução do algoritmo pode não retornar um bom resultado. Portanto, uma maneira comum de implementar o *k-means* é executando-o várias vezes com diferentes configurações de

parâmetros e selecionar o melhor resultados dentre essas execuções (WANG *et al.*, 2017) (note que isso não garante um agrupamento com valor ótimo).

Neste trabalho, adotamos uma pontuação de agrupamento muito utilizada na literatura, denominada *pontuação silhueta*, para medir a qualidade do agrupamento. As definições a seguir foram retiradas de Rousseeuw (1987) (apesar de haver trabalhos mais recentes sobre assunto, como o de Wang *et al.* (2017), esses trabalhos não continham todas as informações necessárias para implementar o algoritmo que calcula a pontuação).

Considere $\mathcal{H} = \{H_1, H_2, \ldots, H_k\}$ um agrupamento. Para todo grupo $H \in \mathcal{H}$ com $|H| > 0$, detonamos por $a(i)$ a distância média entre um objeto (proteína no caso deste trabalho) i e os outros objetos de H, sendo que:

se $|H| \geq 2$, então:

$$a(i) = \frac{\sum_{u \in H \setminus \{i\}} dist(i, u)}{|H| - 1}$$

caso contrário:

$$a(i) = 0$$

Agora, denotemos por $d(i, H)$ a distância média entre i e um grupo $H \in \mathcal{H} \mid i \notin H$:

$$d(i, H) = \frac{\sum_{j \in H} dist(i, j))}{|H|}$$

Designemos por $b(i)$ a distância média entre i e o grupo mais próximo ao seu, ou seja:

$$b(i) = \min_{H \in \mathcal{H} | i \notin H} d(i, H)$$

A pontuação silhueta $ps(i)$ de cada objeto i no conjunto de dados V é obtida através da seguinte fórmula:

$$ps(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Por fim, a pontuação silhueta de um agrupamento $ps(\mathcal{H})$ é a média das pontuações silhuetas de todos $i \in V$:

$$ps(\mathcal{H}) = \frac{\sum_{i \in V} ps(i)}{|V|}$$

Note que a pontuação silhueta de um agrupamento $\mathcal{H}$ só pode ser obtida se $|\mathcal{H}| \geq 2$.

Para gerar o agrupamento de proteínas utilizados nos testes computacionais deste trabalho, executamos diversas vezes o Algoritmo 6 para diferentes valores de k e escolhemos o agrupamento que obteve a maior pontuação silhueta. O melhor agrupamento de acordo com os critérios apresentados possui 1210 grupos, portanto esse foi o número de cores utilizadas em nossos experimentos.

Em alguns casos, a sequência de aminoácidos que descreve uma proteína representada por um vértice $u \in V$ não está contida nos dados. Neste caso, o vértice u fica sem cor ($\xi(u) = \emptyset$) e não é adicionado a nenhum dos grupos. Tais vértices não são removidos da rede, pois os mesmos podem ser utilizados como vértices extras nas ocorrências aproximadas de motivos.

6 Experimentos computacionais

Neste capítulo, apresentamos os resultados dos experimentos computacionais realizados para avaliar empiricamente as soluções propostas neste trabalho para os problemas PBM-CM e PEMM, utilizando dados biológicos reais de redes IPP.

6.1 Metodologia

As instâncias utilizadas foram geradas conforme descrito no Capítulo 5. Utilizamos um arquivo referente à espécie Homo Sapiens (HS), cuja rede IPP possui aproximadamente 8.000 proteínas e 29.000 interações entre elas, e um outro arquivo contendo 968 motivos a serem buscados ou enumerados nessa rede. Após a remoção das arestas com probabilidade menor que 0.5, bem como dos vértices de grau zero, conforme descrito no Capítulo 5, a rede resultante tem 3.055 proteínas e 4.188 interações entre elas. Dentre os motivos presentes nos dados, apenas 678 foram considerados, pois a rede não contém nenhuma ocorrência dos demais motivos.

Todos os códigos foram implementados em linguagem Java, versão 17. O resolvedor de programas lineares inteiros utilizado foi o CPLEX 12.8. Não foi computado o tempo para a leitura dos arquivos das instâncias. Portanto, os tempos mostrados nas tabelas da Seção 6 são referentes apenas à execução dos procedimentos de resolução das respectivas instâncias.

As configurações do ambiente computacional são as seguintes: sistema operacional Windows 10 Pro de 64 bits, processador Intel(R) Core(TM) i5-5200U, CPU 2.20GHz e 16GB de memória RAM.

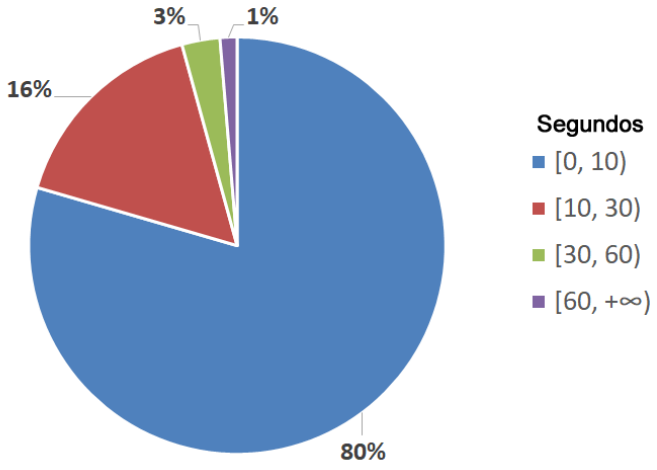
6.2 Análise dos resultados

6.2.1 Problema de busca de motivos de cardinalidade mínima

Denotamos por B&C o método *branch-and-cut* implementado para resolver o modelo $F_{\text{PBMG-CM}}$, conforme descrito no Capítulo 4, ou seja, para o problema em que é retornada apenas a ocorrência de cardinalidade minimal de cada motivo; e cuja rede de entrada é representada por um grafo arbitrário.

Na Figura 23, apresentamos um gráfico relacionando o número de instâncias resolvidas em certos intervalos de tempo. Como é possível perceber, 80% das instâncias foram resolvidas em menos de 10 segundos e somente 1% demorou mais de um minuto para ser solucionada. O tempo médio para resolução das instâncias foi de aproximadamente 9 segundos.

Figura 23 – Percentual de instâncias do PBM-CM resolvidas em função de intervalos de tempo em segundos



Fonte: Ricardo Molinari dos Prazeres, 2022

Na Figura 24, apresentamos um gráfico relacionando o tempo médio de execução de B&C, no eixo vertical, às cardinalidades dos motivos buscados, no eixo horizontal. Entre parênteses, ao lado da cardinalidade do motivo, está a quantidade de motivos buscados com a respectiva cardinalidade.

Apesar de haver oscilação no tempo médio de execução das instâncias à medida em que aumenta a cardinalidade do motivo buscado, nota-se que há uma tendência de aumento desse tempo. Porém, embora não seja um dado de entrada, é possível notar na Figura 25 que o tempo médio de resolução está mais relacionado ao valor ótimo da solução encontrada do que à cardinalidade do motivo buscado.

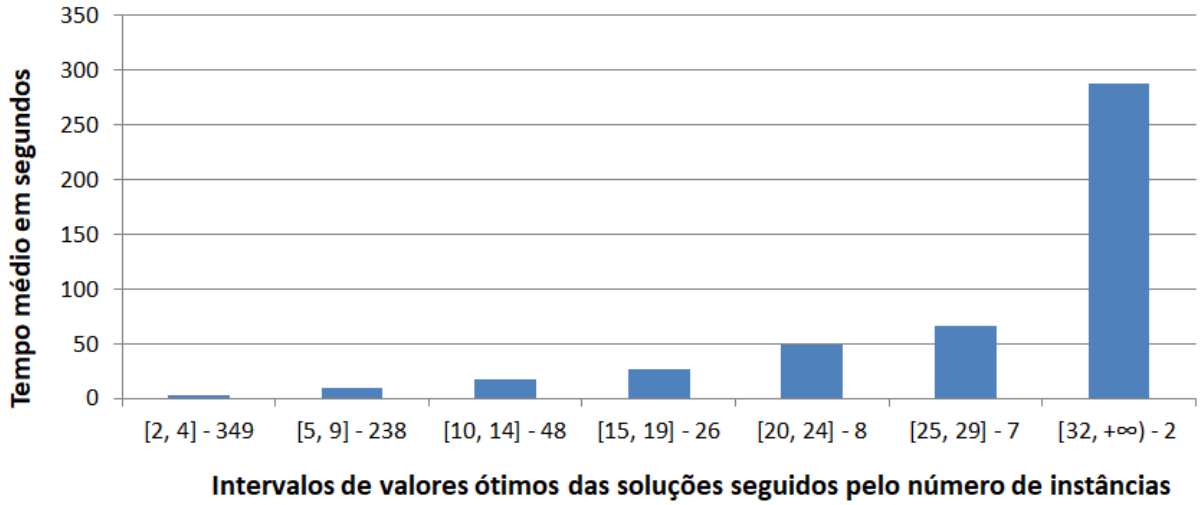
Na Tabela 3, apresentamos informações detalhadas sobre a execução de B&C para as instâncias mais difíceis (as que consumiram mais tempo de execução). As colunas "#Cortes" e "#Lazy" indicam, respectivamente, o número de planos de corte e *lazy constraints* adicionados durante a execução. Alguns nomes de motivos foram abreviados.

Perceba que a instância mais difícil foi resolvida em aproximadamente 8 minutos. Apesar de que o número de inequações do tipo (15) contidas no modelo $F_{\text{PBMG-CM}}$ ser

Figura 24 – Tempo médio de execução de B&C em função da cardinalidade do motivo buscado

Fonte: Ricardo Molinari dos Prazeres, 2022

Figura 25 – Tempo médio de execução de B&C em função do valor ótimo da solução



Fonte: Ricardo Molinari dos Prazeres, 2022

exponencial no tamanho da entrada ($|V| + |A| + |\mathcal{M}|$), através do método *branch-and-cut* apenas um número relativamente pequeno dessas inequações foram adicionadas durante a execução (4123 planos de corte e 584 *lazy constraints*, no pior caso). Também é possível notar que em diversas instâncias, como as de Id 4, 6, 7 e 15, a cardinalidade dos motivos é relativamente baixa, mostrando que esse valor não está tão associado à dificuldade

Tabela 3 – Execução de B&C para as instâncias mais difíceis

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
1	PA700-20S-PA28	11	40	4123	584	8m9s
2	CF IIAm complex	11	29	2121	602	4m43s
3	FA complex	9	24	3637	260	3m29s
4	TRF1-TIN2	3	7	1526	536	1m30s
5	LARC complex	12	32	423	230	1m26s
6	PA28-20S	5	19	490	616	1m13s
7	Profilin 1 complex	4	12	1078	294	1m9s
8	CEN complex	9	19	1657	506	1m5s
9	HDAC1-associated	7	16	655	218	1m2s
10	BRG1-SIN3A	8	25	431	138	55s
11	N-CoR complex	6	7	1110	128	48s
12	BRM-SIN3A-HDAC	8	25	431	138	46s
13	HCF-1 complex	11	17	652	376	46s
14	GPR56-CD81-Galphaq	6	18	866	302	44s
15	Ubiquitin E3	5	10	477	208	41s

Fonte: Ricardo Molinari dos Prazeres, 2022

de resolução da instância, conforme havíamos sugerido quando analisamos o gráfico da Figura 24.

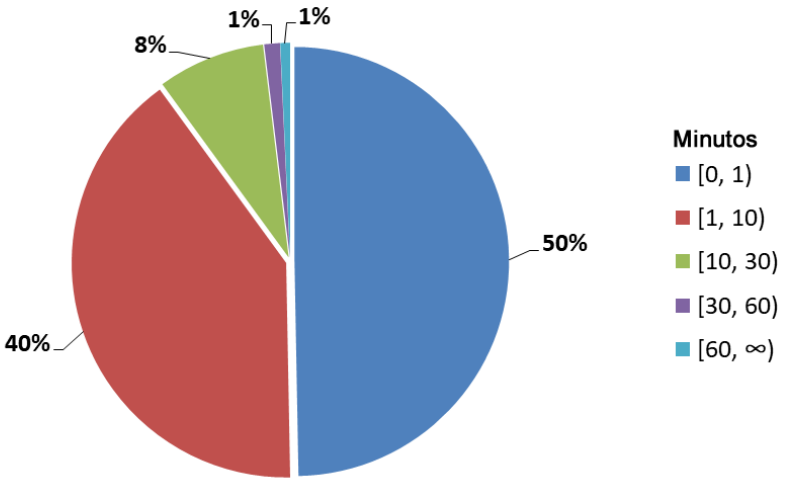
6.2.2 Problema de enumeração de motivos minimais

Denotamos por ALG_{PEMM} o algoritmo de enumeração apresentado na Seção 4.3, para resolver o PEMM. Na Figura 26, apresentamos um gráfico relacionando o número de instâncias resolvidas em certos intervalos de tempo.

Como é possível perceber, apesar de ser um problema de enumeração, metade das instâncias foram resolvidas em menos de 1 minuto e somente 1% demorou mais de 1 hora para ser solucionada. O tempo médio para resolução das instâncias foi de aproximadamente 4 minutos. O número médio de ocorrências encontradas por instância é de aproximadamente $5,5$.

Na Tabela 4, apresentamos informações detalhadas sobre a execução de ALG_{PEMM} para as instâncias mais difíceis do PEMM (as que levaram mais tempo para encontrar todas as ocorrências). A coluna "Nºoc" representa o número de ocorrências minimais de $\mathcal{M}$ contidas em G. As colunas "1º" e "Max" correspondem à primeira ocorrência e a ocorrência com cardinalidade máxima (ou com tempo máximo de execução), respectivamente. A

Figura 26 – Percentual de instâncias do PEMM resolvidas em função de intervalos de tempo em minutos



Fonte: Ricardo Molinari dos Prazeres, 2022

coluna “AVG” corresponde à cardinalidade média (ou tempo médio de execução) entre todas as ocorrências. A coluna “Total” corresponde ao tempo total consumido.

Nota-se que, na Tabela 4, em todos os casos o tempo médio gasto para encontrar uma ocorrência é maior que o tempo consumido para encontrar a primeira ocorrência. Ou seja, à medida em que adicionamos restrições do tipo (16), que impedem que os motivos já listados sejam enumerados novamente, o problema se torna mais difícil de ser resolvido (pelo menos até um certo momento). A ocorrência mais difícil (levando em conta o tempo para ser solucionada) pertence à instância de Id 2, levando cerca de 1 hora e 33 minutos para ser resolvida. Já na instância mais difícil (Id 1), a ocorrência mais demorada levou cerca de 25 minutos para ser resolvida, porém essa instância teve um número bem maior de ocorrências (9 no total) que a de Id 2 (3 no total), o que ocasionou a maior quantidade de tempo.

A seguir, apresentamos a Tabela 5, na qual estão informações sobre todas as ocorrências das 3 instâncias mais difíceis da Tabela 4.

Em relação à Tabela 5, nota-se que nas 3 instâncias a cardinalidade da última ocorrência encontrada (que tem cardinalidade máxima) é consideravelmente maior do que a do primeiro (que tem cardinalidade mínima). No motivo *COG complex*, por exemplo, tais ocorrências têm cardinalidade 16 e 41, respectivamente. Mas, mesmo com a cardinalidade das ocorrências aumentando, nos motivos *COG complex* e *P2X7 receptor*, o tempo de solução das ocorrências aumenta até atingir um determinado pico (Id 7 e 18),

Tabela 4 – Execução de ALG_{PEMM} para as instâncias mais difíceis do PEMM

			Cardinalidade			Consumo de tempo			
Id	Motivo	NºOc	1°	Avg	Máx	1°	Avg	Máx	Total
1	COG complex	9	16	23	27	17s	13m27s	25m31s	2h1m
2	PA700-20S-PA28	3	40	43	50	8m9s	36m11s	1h33m38s	1h49m
3	P2X7 receptor	8	14	18	21	41s	9m18s	23m17s	1h14m
4	ITGA2b-ITGB3	11	11	17	25	18s	6m19s	14m23s	1h9m
5	BRCA1-RNA	5	19	25	37	22s	13m45s	44m12s	1h9m
6	BARD1-BRCA1	16	5	12	18	21s	3m27s	14m18s	55m
7	CF IIAm complex	4	29	31	38	4m43s	12m44s	29m5s	51m
8	DGCR8	8	15	20	30	36s	5m29s	13m24s	44m
9	CSA-POLIIa	9	8	16	20	11s	4m4s	9m19s	37m
10	RalBP1-CCNB1	6	13	16	18	31s	6m4s	14m44s	36m
11	p27-cyclinE	8	9	18	20	9s	4m6s	8m6s	33m
12	ITGAV-ITGB8	9	7	14	15	14s	3m32s	8m52s	32m
13	R/M complex	30	2	7	18	11s	56s	3m18s	28m
14	HCF-1 complex	5	17	20	26	46s	5m16s	15m38s	26m
15	Ubiquitin E3	7	10	16	17	41s	3m44s	8m23s	26m

Fonte: Ricardo Molinari dos Prazeres, 2022

e depois esse tempo começa a diminuir. Apesar desse comportamento não ocorrer no motivo *PA700-20S-PA28*, ele ocorre na maioria das instâncias (as demais instâncias presentes na Tabela 4 podem ser visualizadas na Tabela 8).

Tabela 5 – Todas as ocorrências das 3 instâncias mais difíceis do PEMM

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
1	COG complex	7	16	169	528	17s
2	COG complex	7	16	251	632	41s
3	COG complex	7	19	1851	1176	2m48s
4	COG complex	7	19	4862	1006	8m55s
5	COG complex	7	21	8515	1590	14m32s
6	COG complex	7	23	11631	1718	22m20s
7	COG complex	7	27	13799	1100	**25m30s**
8	COG complex	7	33	17217	1336	23m33s
9	COG complex	7	41	14597	820	22m26s
10	PA700-20S-PA28	11	40	4123	584	8m9s
11	PA700-20S-PA28	11	40	4165	1032	6m45s
12	PA700-20S-PA28	11	50	19159	1040	**1h33m37s**
13	P2X7 receptor	6	14	880	550	41s
14	P2X7 receptor	6	14	1353	760	55s
15	P2X7 receptor	6	16	2602	430	3m34s
16	P2X7 receptor	6	17	5672	482	9m28s
17	P2X7 receptor	6	18	5567	480	8m28s
18	P2X7 receptor	6	21	11420	444	**23m16s**
19	P2X7 receptor	6	24	12445	314	21m1s
20	P2X7 receptor	6	24	8023	196	6m53s

Fonte: Ricardo Molinari dos Prazeres, 2022

6.3 Análise das ocorrências aproximadas

Como mencionado anteriormente, há diferentes definições de ocorrências aproximadas de motivos. Nesta seção, realizamos experimentos para tentar medir a "qualidade" das ocorrências aproximadas retornadas como solução do PBM-CM e discutimos alguns aspectos relacionados ao PEMM e ocorrências aproximadas.

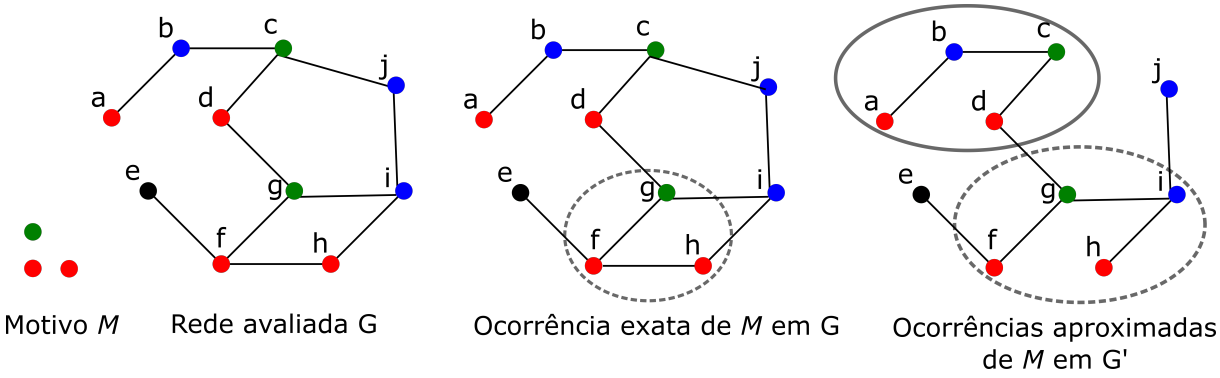
6.3.1 Objetivo do experimento

Também como já mencionado em outros capítulos, o interesse em encontrar ocorrências aproximadas de um motivo, caso não haja exatas, deve-se ao fato de que pode haver falhas durante o processo de obtenção dos dados biológicos, e por consequência não é possível afirmar com certeza a existência ou ausência de cada aresta da rede. Isso pode resultar em casos em que um determinado motivo ocorre de modo exato no organismo

em questão, mas a rede que o representa contém apenas ocorrências aproximadas desse motivo.

Considere G o grafo que representa uma dada rede biológica e G' a rede obtida ao remover-se uma quantidade n de arestas de G. O que propomos nesta seção é verificar o númeno de ocorrências aproximadas encontradas em G' que são ocorrências exatas em G (ocorrências exatas contidas nas aproximadas). A Figura 27 ilustra a ideia apresentada acima, sendo que a rede G' é obtida removendo-se as arestas $\{cj, fh\}$ de G.

Figura 27 – Exemplos de ocorrências aproximadas que contêm e que não contêm uma ocorrência exata



Fonte: Ricardo Molinari dos Prazeres, 2022

Na Figura 27, podemos visualizar (elipse pontilhada em G) a ocorrência exata $E = \{f, g, h\}$ de M em G. Ainda nessa figura, na rede G' é possível verificar as ocorrências aproximadas $P = \{f, g, h, i\}$ (elipse pontilhada em G') e $L = \{a, b, c, d\}$ (elipse contínua), sendo que $E \subset P$ e $E \not\subset L$.

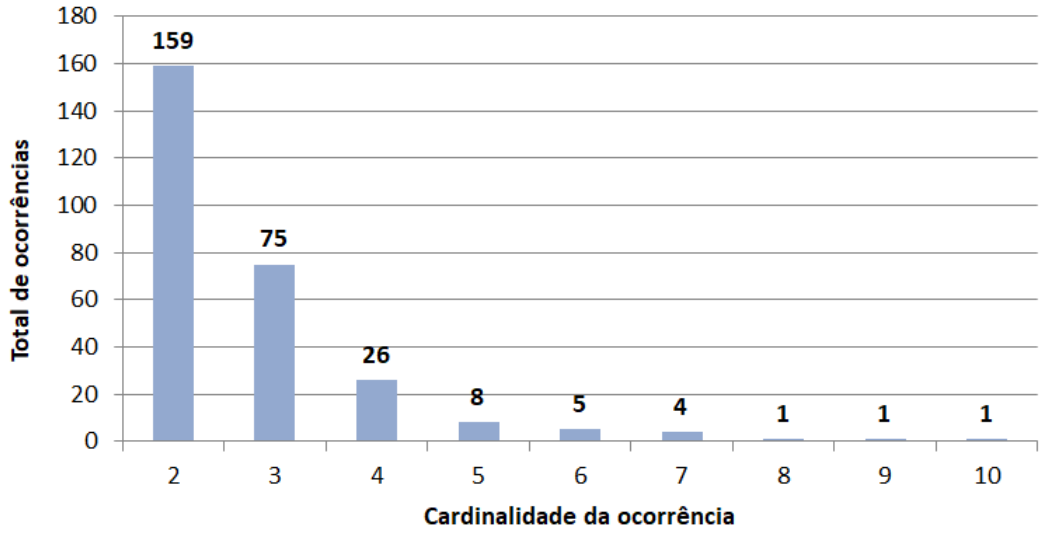
O objetivo do experimento realizado nesta seção é analisar os dados referentes ao número de ocorrências aproximadas de M em G' que contenham e que não contenham as ocorrências exatas de M em G.

6.3.2 Instâncias e resultados

Todos os experimentos realizados nesta seção envolvem o problema de busca de motivos proposto neste trabalho, ou seja o PBM-CM. No final da seção, comentamos sobre o comportamento do problema de enumeração de motivos minimais (PEMM) em relação a ocorrências aproximadas.

Para a instância de motivos deste experimento, utilizamos todos os motivos nos quais foram encontradas ocorrências exatas pelo B&C (algoritmo do PBM-CM) nos experimentos realizados na Seção 6.2.1. Há um total de 280 motivos desse tipo, e suas cardinalidades variam de 2 a 10, na proporção mostrada na Figura 28.

Figura 28 – Número de ocorrências por cardinalidade



Fonte: Ricardo Molinari dos Prazeres, 2022

Para as instâncias de rede biológica deste experimento, modificamos a rede original removendo aleatoriamente um certo número de suas arestas. Utilizamos 5 redes modificadas, variando entre elas o percentual de arestas removidas em 10, 20, 30, 40 e 50%.

Escolhemos arbitrariamente a rede modificada, que denominaremos por $G10$, com percentual de remoção de aresta de 10% para exibir e discutir sobre alguns dados inciais nas soluções encontradas por B&C em $G10$. Referenciaremos a rede original por G.

Na Tabela 6 escolhemos alguns motivos, com cardinalidade diferentes entre eles, para discutir algumas pontos. A coluna “Ocor exata contida” indica se a ocorrência do motivo encontrada em $G10$ contém a ocorrência exata encontrada em G.

Podemos notar na Tabela 6 que no motivo de Id 1, mesmo esse tendo cardinalidade relativamente alta, a ocorrência aproximada encontrada contém a ocorrência exata da rede original. Nos motivos de Id 2 e 4, a cardinalidade dos motivos e valor das soluções ótimas são iguais, como essas soluções são exatas, podemos deduzir que são as mesmas nas duas redes, ou seja, não sofreram interferência do processo de remoção de arestas. No motivo de Id 3, vemos como a remoção de arestas pode gerar impactos significativos no tamanho da

Tabela 6 – Verificação se ocorrências encontradas na rede modificada contêm ocorrências exata na rede original

Id	Motivo	$\lvert\mathcal{M}\rvert$	$\lvert H^*\rvert$	Ocorrência exata contida
1	DAB complex	10	11	Sim
2	DA complex	9	9	Sim
3	COG complex	7	17	Não
4	CSA complex	6	6	Sim
5	LAT-PLC-gamma-1-p85-GRB2-SOS	5	6	Não
6	JUND-FOSB-SMAD3-SMAD4	4	4	Não
7	BRCA1-BARD1-POLR2A	3	5	Sim
8	NELF complex	2	6	Sim

Fonte: Ricardo Molinari dos Prazeres, 2022

solução. Em relação ao motivo de Id 6, temos um caso em que há uma ocorrência exata, porém essa não contém a ocorrência exata em G, sendo portanto diferentes. Isso pode acontecer porque a remoção de arestas fez com a ocorrência exata de G precisasse de pelo menos mais um vértice para manter-se conexa ou apenas que, por se tratar de uma rede diferente, o resolvedor "escolheu" outra solução de mesma cardinalidade. Já no motivo de Id 8, mesmo a ocorrência possuindo uma cardinalidade relativamente alta em relação à do motivo, foi possível encontrar uma solução que contém a ocorrência exata em G. Por fim, apesar de haver ocorrências exatas de motivos com cardinalidade igual a 8 em G, não foi possível encontrar nenhuma ocorrência, mesmo aproximada, em $G10$, o que mostra que $G10$ não é conexa, e nenhuma de suas componentes contêm ocorrências desses motivos.

A Tabela 7 contém uma sumarização das informações referentes à execução do algoritmo B&C em cada uma das redes modificadas. Executamos o algoritmo 10 vezes em cada rede com percentual de remoção distinto, sendo os valores apresentamos a média dos valores das 10 execuções. A coluna "%" indica o percentual de arestas removidas daquela rede. A coluna "Total" representa o total de ocorrências encontradas, lembrando que o total de motivos buscados é 280. Em "Cont" exibimos o número de ocorrências exatas encontradas na rede original que estão contidas na solução da rede modificada; a mesma ideia se aplica em "ÑCont". A coluna "Iguais" indica o total de ocorrências exatas que são exatamente as mesma na rede original e na modificada, a mesma ideia se aplica em "Diferentes". Como nosso objetivo é analisar ocorrências aproximadas, criamos mais duas colunas para esse propósito: "ACont" e "AÑCont", a primeira é obtida subtraindo-se

Tabela 7 – Valores médios de ocorrências contidas e não contidas referentes à execução de B&C em diferentes redes modificadas

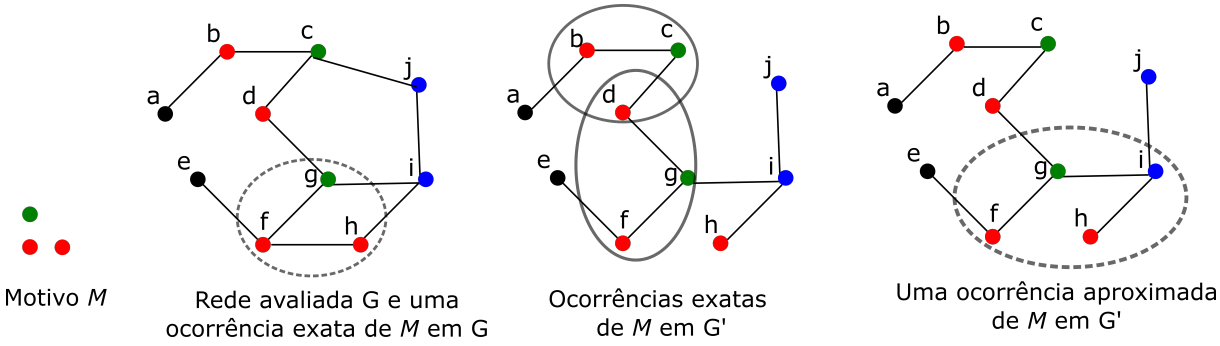
				Exatas		Aproximadas		
%	Total	Cont	ÑCont	Iguais	Diferentes	ACont	AÑCont	Razão
10	273,8	236,2	37,6	226,1	15,6	10,1	22,0	0,53
20	265,4	200,2	65,2	180,7	30,8	19,5	34,4	0,59
30	253,3	171,9	81,4	149,6	29,3	22,3	52,1	0,48
40	237,2	142,6	94,6	117,8	40,6	24,8	54,0	0,46
50	216,2	117,4	98,8	94,2	33,1	23,2	65,7	0,35

Fonte: Ricardo Molinari dos Prazeres, 2022

"Iguais" de "Cont" e a segunda "Diferentes" de "ÑCont". Por fim, a coluna "Razão" indica a razão entre "ACont" e "AÑCont".

Na Tabela 7, podemos notar que, conforme esperávamos, o número médio de ocorrências encontradas em cada rede diminuiu conforme o percentual de arestas removidas aumenta. Notamos também que, de modo geral, o número de ocorrências aproximadas aumentam, o que provavelmente indica que as arestas que tornavam possível a existência de certas ocorrências exatas foram removidas. Agora em relação à razão entre as ocorrências aproximadas que contêm e que não contêm ocorrências exatas: exceto na rede com percentual de remoção de aresta de 50%, a razão fica por volta de 0.5, ou seja, aproximadamente a cada 3 ocorrências aproximadas encontradas, uma contém uma ocorrência exata. Apresentamos na Tabela 9 os dados de cada uma das 10 execuções de cada rede.

Figura 29 – Ocorrências minimais encontradas no PEMM



Fonte: Ricardo Molinari dos Prazeres, 2022

Por fim, em relação ao PEMM: como nesse problema são retornadas todas as ocorrências minimais de um motivo, caso uma ocorrência exata E na rede original ainda

esteja contida de forma conexa na rede modificada, o algoritmo desse problema encontrará E, de modo exato ou aproximado. A Figura 29 mostra um exemplo em que uma determinada ocorrência exata $E = \{f, g, h\}$ de $\mathcal{M}$ em G (elipse pontilhada em G) não pode mais ser encontrada de forma exata em G'. Note que mesmo havendo outras ocorrências minimais de $\mathcal{M}$ em G' (elipses contínuas), em algum momento o algoritmo encontrará uma ocorrência aproximada que contenha E (elipse pontilhada em G'), caso ainda esteja contida de forma conexa em G'.

7 Conclusão e trabalhos futuros

Neste trabalho, estudamos os problemas de busca e enumeração de motivos em grafos, que possuem aplicações na área de biologia. Propusemos uma abordagem *branch-and-cut* para resolver o problema de busca e, através de uma pequena adaptação, propusemos um algoritmo para resolver o problema de enumeração. Nossa abordagem difere das encontradas na literatura nos seguintes aspectos: com relação ao problema de busca, o modelo de PI proposto anteriormente só resolve o caso específico em que o grafo é acíclico (árvore), enquanto propomos 3 modelos, entre eles um que resolve o caso geral do problema (grafos arbitrários); com relação ao problema de enumeração, o algoritmo proposto anteriormente utiliza uma abordagem de força bruta, em que nos experimentos computacionais enumeram apenas motivos com cardinalidade de no máximo 4, enquanto que a nossa abordagem mostrou-se capaz de enumerar motivos com dezenas de proteínas. No problema de busca, conseguimos resolver a maioria das instâncias em poucos segundos, em alguns minutos as mais difíceis. Já no problema de enumeração, foi possível solucionar a maior parte das instâncias em menos de 1 hora, sendo que a mais difícil demorou 2 horas para ser resolvida.

Para realização do agrupamento em cores das proteínas da instância de rede IPP utilizada em nossos experimentos computacionais, definimos uma pontuação de semelhança entre as proteínas baseada no valor da subsequência comum mais longa entre os aminoácidos dessas proteínas; implementamos uma adaptação do algoritmo *k-means* para agrupar as proteínas de acordo com a pontuação de semelhança entre elas; implementamos um método de avaliação da qualidade do agrupamento, denominado pontuação silhueta, e executamos o algoritmo de agrupamento diversas vezes até encontrar o que recebeu a melhor avaliação.

Como trabalhos futuros, sugerimos abordar esses problemas em hipergrafos. Em Andrade *et al.* (2015), ou autores citam algumas aplicações na área de biologia nas quais as redes em questão são representadas por hipergrafos.

Referências

ANDRADE, R.; BIRMELÉ, E.; MARY, A.; PICCHETTI, T.; SAGOT, M.-F. Incremental complexity of a bi-objective hypergraph transversal problem. In: *Fundamentals of Computation Theory*. Cham: Springer International Publishing, 2015. p. 202–213. ISBN 978-3-319-22177-9. Citado na página 69.

BACKES, C.; RURAINSKI, A.; KLAU, G.; MÜLLER, O.; STÖCKEL, D.; GERASCH, A.; KÜNTZER, J.; MAISEL, D.; LUDWIG, N.; HEIN, M.; KELLER, A.; BURTSCHER, H.; KAUFMANN, M.; MEESE, E.; LENHOF, H.-P. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Research*, v. 40, 12 2011. Citado 2 vezes nas páginas 34 e 36.

BERGROTH, L.; HAKONEN, H.; RAITA, T. A survey of longest common subsequence algorithms. In: *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, 2000. p. 39–48. Citado na página 52.

BOLLOBÁS, B. *Modern Graph Theory*. 1. ed.: Springer-Verlag New York, 1998. (Graduate Texts in Mathematics 184). ISBN 978-0-387-98488-9,978-1-4612-0619-4. Citado na página 19.

BONDY, J.; MURTY, U. *Graph Theory*: Springer, 2008. Citado na página 19.

BONDY, J. A.; MURTY, U. S. R. Book. *Graph Theory with Applications / J. A. Bondy and U. S. R. Murty*: Macmillan London, 1976. ISBN 0333177916. Citado na página 19.

BRUCKNER, S.; HÜFFNER, F.; KARP, R.; SHAMIR, R.; SHARAN, R. Topology-free querying of protein interaction networks. *Journal of Computational Biology : a Journal of Computational Molecular Cell Biology*, v. 17, p. 237–252, 03 2010. Citado 3 vezes nas páginas 34, 35 e 36.

CARVAJAL, R.; CONSTANTINO, M.; GOYCOOLEA, M.; VIELMA, J.; WEINTRAUB, A. Imposing connectivity constraints in forest planning models. *Operations Research*, v. 61, p. 824–836, 08 2013. Citado 3 vezes nas páginas 16, 45 e 47.

CHOPRA, S.; FILIPECKI, B.; LEE, K.; RYU, M.; SHIM, S.; VYVE, M. An extended formulation of the convex recoloring problem on a tree. *Math. Program.*, Springer-Verlag, Berlin, Heidelberg, v. 165, n. 2, p. 529–548, oct 2017. ISSN 0025-5610. Disponível em: ⟨https://doi.org/10.1007/s10107-016-1094-3⟩. Citado na página 44.

CIRIELLO, G.; GUERRA, C. A review on models and algorithms for motif discovery in protein–protein interaction networks. *Briefings in Functional Genomics*, v. 7, n. 2, p. 147–156, 04 2008. ISSN 2041-2649. Disponível em: ⟨https://doi.org/10.1093/bfgp/eln015⟩. Citado na página 36.

CONFORTI, M.; CORNUEJOLS, G.; ZAMBELLI, G. *Integer Programming*: Springer Publishing Company, Incorporated, 2014. ISBN 3319110071. Citado 5 vezes nas páginas 24, 26, 28, 29 e 31.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms, Third Edition*. 3rd. ed.: The MIT Press, 2009. ISBN 0262033844. Citado 4 vezes nas páginas 22, 23, 24 e 49.

DINH, C. H.; RAJASEKARAN, S.; KUNDETI, V. PMS5: An efficient exact algorithm for the (l, d)-motif finding problem. *BMC Bioinformatics*, v. 12, 10 2011. Citado 3 vezes nas páginas 33, 34 e 36.

DITTRICH, M.; KLAU, G.; ROSENWALD, A.; DANDEKAR, T.; MÜLLER, T. Identifying functional modules in protein-protein interaction networks: An integrated exact approach. *Bioinformatics (Oxford, England)*, v. 24, 08 2008. Citado 2 vezes nas páginas 34 e 36.

EDLA, D. R.; TRIPATHI, D.; KUPPILI, V.; CHERUKU, R. Survey on clustering techniques. In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018. p. 696–703. Citado na página 52.

FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. *Uma Introdução Sucinta à Teoria dos Grafos*, 2011. Disponível em: ⟨https://www.ime.usp.br/~pf/teoriadosgrafos/texto/TeoriaDosGrafos.pdf⟩. Citado 2 vezes nas páginas 19 e 20.

FREIRE, A. da S.; LIMA, K. R. P. S.; ROJAS, D. I. Z. A system for motif search in biological networks. In: *Proceedings of the XIV Brazilian Symposium on Information Systems*. New York, NY, USA: Association for Computing Machinery, 2018. (SBSI'18). ISBN 9781450365598. Disponível em: ⟨https://doi.org/10.1145/3229345.3229378⟩. Citado 9 vezes nas páginas 15, 16, 17, 18, 34, 35, 36, 38 e 44.

GRÖTSCHEL, M.; LÁSZLÓ, L.; SCHRIJVER, A. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, v. 1, p. 169–197, 06 1981. Citado na página 30.

HASHIM, F. A.; MABROUK, M. S.; AL-ATABANY, W. Review of different sequence motif finding algorithms. *Avicenna Journal of Medical Biotechnology*, v. 11, n. 1-2, p. 130–148, 2019. Citado 2 vezes nas páginas 32 e 33.

HATTORI, L.; GUTOSKI, M.; BENÍTEZ, C.; NUNES, L.; LOPES, H. A benchmark of optimally folded protein structures using integer programming and the 3d-hp-sc model. *Computational Biology and Chemistry*, v. 84, 12 2019. Citado na página 34.

KINGSFORD, C.; ZASLAVSKY, E.; SINGH, M. A cost-aggregating integer linear program for motif finding. *Journal of Discrete Algorithms*, v. 9, n. 4, p. 326–334, 2011. ISSN 1570-8667. Selected papers from the 17th Annual Symposium on Combinatorial Pattern Matching (CPM 2006). Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S157086671100044X⟩. Citado 3 vezes nas páginas 33, 34 e 36.

LACROIX, V.; FERNANDES, C.; SAGOT, M.-F. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics / IEEE, ACM*, v. 3, p. 360–368, 11 2006. Citado 8 vezes nas páginas 15, 16, 17, 18, 34, 36, 38 e 41.

LEGENDRE, A.; ANGEL, E.; TAHI, F. Bi-objective integer programming for rna secondary structure prediction with pseudoknots. *BMC Bioinformatics*, v. 19, 01 2018. Citado na página 34.

LIMA, K.; WAKABAYASHI, Y. Convex recoloring of paths. *Electronic Notes in Discrete Mathematics*, v. 37, p. 165–170, 08 2011. Citado na página 44.

LIMA, K. R.; BRIGATTO, F. Desenvolvimento de uma formulação inteira para o problema de motifs em grafos. In: *Anais do XI Simpósio Brasileiro de Sistemas de Informação - II Workshop de Iniciação Científica de Sistemas de Informação WICSI*. Goiás, GO, Brasil: SBC, 2015. v. 2, p. 29–32. Disponível em: ⟨http://www2.sbc.org.br/ce-si/arquivos/anais/SBSI2015-Anais-WICSI.pdf⟩. Citado 8 vezes nas páginas 15, 16, 18, 34, 35, 36, 38 e 40.

MAKOLO, A. A comparative analysis of motif discovery algorithms. *Computational Biology and Bioinformatics*, v. 4, n. 1-2, p. 1–9, 2016. Citado na página 33.

MALIK, S.; SHARMA, D. Detecting history of species using mining of motifs in phylogenetic networks. In: *Proceedings of the 2014 International Conference on Information and Communication Technology for Competitive Strategies*. New York, NY, USA: ACM, 2014. (ICTCS '14). ISBN 978-1-4503-3216-3. Disponível em: ⟨http://doi.acm.org/10.1145/2677855.2677935⟩. Citado na página 15.

MAZZA, A.; KLOCKMEIER, K.; WANKER, E.; SHARAN, R. An integer programming framework for inferring disease complexes from network data. *Bioinformatics*, v. 32, 12 2016. Citado 2 vezes nas páginas 34 e 36.

POOLSAP, U.; KATO, Y.; AKUTSU, T. Prediction of rna secondary structure with pseudoknots using integer programming. *BMC Bioinformatics*, v. 10 Suppl 1, 02 2009. Citado na página 34.

ROUSSEEUW, P. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, v. 20, p. 53–65, 11 1987. Citado na página 55.

SANDVE, G.; DRABLOS, F. A survey of motif discovery methods in an integrated framework. *Biology Direct*, v. 1, 02 2006. Citado na página 33.

SHYU, S. J.; TSAI, C.-Y. Finding the longest common subsequence for multiple biological sequences by ant colony optimization. *Computers Operations Research*, v. 36, n. 1, p. 73–91, 2009. ISSN 0305-0548. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0305054807001347⟩. Citado na página 51.

TANG, H.; LYONS, E.; PEDERSEN, B.; SCHNABLE, J.; PATERSON, A.; FREELING, M. Screening synteny blocks in pairwise genome comparisons through integer programming. *BMC Bioinformatics*, v. 12, 04 2011. Citado 3 vezes nas páginas 33, 34 e 36.

WANG, F.; FRANCO-PENYA, H.-H.; KELLEHER, J. D.; PUGH, J.; ROSS, R. An analysis of the application of simplified silhouette to the evaluation of k-means clustering validity. In: PERNER, P. (Ed.). *Machine Learning and Data Mining in Pattern Recognition*. Cham: Springer International Publishing, 2017. p. 291–305. ISBN 978-3-319-62416-7. Citado na página 55.

YANG, S.; TOWEY, D.; ZHOU, Z. Q. Metamorphic exploration of an unsupervised clustering program. In: *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*, 2019. p. 48–54. Citado na página 53.

YU, Q.; HUO, H.; ZHANG, Y.; GUO, H. Pairmotif: A new pattern-driven algorithm for planted (l, d) dna motif search. *PloS One*, v. 7, 10 2012. Citado 3 vezes nas páginas 33, 34 e 36.

ZASLAVSKY, E.; SINGH, M. A combinatorial optimization approach for diverse motif finding applications. *Algorithms for Molecular Biology: AMB*, v. 1, 02 2006. Citado 3 vezes nas páginas 33, 34 e 36.

Apêndice A – Mais resultados referentes ao experimentos computacionais

A seguir, apresentamos a Tabela 8, na qual estão informações sobre todas as ocorrências das 15 instâncias mais difíceis do PEMM. Em negrito estão o maior tempo gasto entre as ocorrências de uma mesma instância, mostrando que na maioria dos casos, mesmo com a cardinalidade da solução aumentando, o tempo diminui após atingir um certo pico.

Tabela 8 – Todas as ocorrências das instâncias mais difíceis do PEMM

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
1	COG complex	7	16	169	528	17s
2	COG complex	7	16	251	632	41s
3	COG complex	7	19	1851	1176	2m48s
4	COG complex	7	19	4862	1006	8m55s
5	COG complex	7	21	8515	1590	14m32s
6	COG complex	7	23	11631	1718	22m20s
7	COG complex	7	27	13799	1100	**25m30s**
8	COG complex	7	33	17217	1336	23m33s
9	COG complex	7	41	14597	820	22m26s
10	PA700-20S-PA28	11	40	4123	584	8m9s
11	PA700-20S-PA28	11	40	4165	1032	6m45s
12	PA700-20S-PA28	11	50	19159	1040	**1h33m37s**
13	P2X7 receptor	6	14	880	550	41s
14	P2X7 receptor	6	14	1353	760	55s
15	P2X7 receptor	6	16	2602	430	3m34s
16	P2X7 receptor	6	17	5672	482	9m28s
17	P2X7 receptor	6	18	5567	480	8m28s
18	P2X7 receptor	6	21	11420	444	**23m16s**
19	P2X7 receptor	6	24	12445	314	21m1s
20	P2X7 receptor	6	24	8023	196	6m53s
21	ITGA2b-ITGB3	3	3	0	92	5s
22	ITGA2b-ITGB3	3	5	128	160	7s

continua

continuação

Id	Motivo	$\lvert\mathcal{M}\rvert$	$\lvert H^*\rvert$	#Cortes	#Lazy	Tempo
23	ITGA2b-ITGB3	3	6	588	268	32s
24	ITGA2b-ITGB3	3	9	2878	762	2m54s
25	ITGA2b-ITGB3	3	9	753	752	48s
26	ITGA2b-ITGB3	3	10	1975	594	1m49s
27	ITGA2b-ITGB3	3	10	1675	578	59s
28	ITGA2b-ITGB3	3	12	2451	360	2m7s
29	ITGA2b-ITGB3	3	12	6414	772	**3m11s**
30	ITGA2b-ITGB3	3	14	3261	1166	1m29s
31	ITGA2b-ITGB3	3	14	4442	540	1m53s
32	ITGA2b-ITGB3	3	17	741	228	26s
33	ITGA2b-ITGB3	3	18	1442	212	38s
34	BRCA1-RNA	13	19	24	172	22s
35	BRCA1-RNA	13	19	31	182	21s
36	BRCA1-RNA	13	21	126	218	26s
37	BRCA1-RNA	13	33	8626	1366	23m23s
38	BRCA1-RNA	13	37	15386	664	**44m11s**
39	BARD1-BRCA1	3	5	447	186	20s
40	BARD1-BRCA1	3	5	84	120	8s
41	BARD1-BRCA1	3	7	707	340	36s
42	BARD1-BRCA1	3	7	761	274	38s
43	BARD1-BRCA1	3	9	1136	490	1m10s
44	BARD1-BRCA1	3	10	3233	746	3m34s
45	BARD1-BRCA1	3	10	3716	678	2m54s
46	BARD1-BRCA1	3	12	4345	694	2m30s
47	BARD1-BRCA1	3	12	2912	1074	2m10s
48	BARD1-BRCA1	3	13	5700	874	4m22s
49	BARD1-BRCA1	3	14	5787	612	2m31s
50	BARD1-BRCA1	3	14	5489	808	4m5s
51	BARD1-BRCA1	3	16	8465	716	5m20s

continua

continuação

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
52	BARD1-BRCA1	3	18	9159	692	**14m17s**
53	BARD1-BRCA1	3	19	4742	716	4m38s
54	BARD1-BRCA1	3	20	2766	962	5m57s
55	CF IIAm complex	11	29	2121	602	4m43s
56	CF IIAm complex	11	29	2170	544	2m26s
57	CF IIAm complex	11	31	5159	464	14m39s
58	CF IIAm complex	11	38	9480	454	**29m5s**
59	DGCR8	8	15	517	424	35s
60	DGCR8	8	15	1229	1236	1m47s
61	DGCR8	8	16	1320	914	1m44s
62	DGCR8	8	18	3137	754	3m59s
63	DGCR8	8	20	4499	818	4m12s
64	DGCR8	8	22	6263	1438	6m15s
65	DGCR8	8	27	11747	550	11m51s
66	DGCR8	8	30	9059	1246	**13m23s**
67	CSA-POLIIa	7	8	29	150	11s
68	CSA-POLIIa	7	11	367	486	22s
69	CSA-POLIIa	7	12	386	272	20s
70	CSA-POLIIa	7	17	2719	498	2m17s
71	CSA-POLIIa	7	17	5344	1502	7m1s
72	CSA-POLIIa	7	19	3194	734	4m18s
73	CSA-POLIIa	7	20	6407	652	**9m18s**
74	CSA-POLIIa	7	24	7283	598	8m44s
75	CSA-POLIIa	7	24	4719	578	4m2s
76	RalBP1-CCNB1	6	13	229	154	31s
77	RalBP1-CCNB1	6	13	341	94	42s
78	RalBP1-CCNB1	6	15	513	154	1m19s
79	RalBP1-CCNB1	6	18	4702	388	12m6s
80	RalBP1-CCNB1	6	18	4828	376	**14m44s**

continua

continuação

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
81	RalBP1-CCNB1	6	20	2758	1358	7m1s
82	p27-cyclinE	7	9	69	116	8s
83	p27-cyclinE	7	9	6	96	6s
84	p27-cyclinE	7	12	840	342	36s
85	p27-cyclinE	7	16	3504	544	3m52s
86	p27-cyclinE	7	19	4571	456	8m4s
87	p27-cyclinE	7	20	6086	444	**8m5s**
88	p27-cyclinE	7	27	4682	574	5m58s
89	p27-cyclinE	7	33	5317	456	5m53s
90	ITGAV-ITGB8	3	7	172	364	13s
91	ITGAV-ITGB8	3	8	447	106	12s
92	ITGAV-ITGB8	3	8	897	180	32s
93	ITGAV-ITGB8	3	12	4857	300	3m38s
94	ITGAV-ITGB8	3	13	4088	194	4m39s
95	ITGAV-ITGB8	3	15	5870	366	**8m51s**
96	ITGAV-ITGB8	3	15	6303	212	7m54s
97	ITGAV-ITGB8	3	22	7888	488	3m4s
98	ITGAV-ITGB8	3	27	5304	240	2m40s
99	R/M complex	2	2	70	440	11s
100	R/M complex	2	3	26	244	6s
101	R/M complex	2	4	114	432	16s
102	R/M complex	2	5	79	364	13s
103	R/M complex	2	5	213	688	21s
104	R/M complex	2	5	231	730	24s
105	R/M complex	2	5	50	582	15s
106	R/M complex	2	5	147	856	23s
107	R/M complex	2	6	171	744	20s
108	R/M complex	2	6	323	708	25s
109	R/M complex	2	6	298	874	27s

continua

continuação

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
110	R/M complex	2	6	177	826	22s
111	R/M complex	2	6	235	838	27s
112	R/M complex	2	6	310	868	27s
113	R/M complex	2	6	419	990	34s
114	R/M complex	2	6	286	918	32s
115	R/M complex	2	7	630	1426	1m20s
116	R/M complex	2	7	334	906	30s
117	R/M complex	2	8	633	1946	1m40s
118	R/M complex	2	9	549	2640	2m25s
119	R/M complex	2	9	1095	2074	1m57s
120	R/M complex	2	9	1834	1978	1m55s
121	R/M complex	2	9	1202	1708	1m25s
122	R/M complex	2	9	1647	1654	1m55s
123	R/M complex	2	10	1109	1948	1m37s
124	R/M complex	2	11	1587	1596	1m8s
125	R/M complex	2	13	541	1622	49s
126	R/M complex	2	14	808	1396	47s
127	R/M complex	2	18	3740	1752	**3m17s**
128	R/M complex	2	21	1992	1280	1m2s
129	HCF-1 complex	11	17	652	376	46s
130	HCF-1 complex	11	17	311	326	30s
131	HCF-1 complex	11	18	1173	330	1m15s
132	HCF-1 complex	11	24	6378	386	8m11s
133	HCF-1 complex	11	25	7774	442	**15m37s**
134	Ubiquitin E3	5	10	477	208	41s
135	Ubiquitin E3	5	11	270	262	43s
136	Ubiquitin E3	5	14	1359	546	1m38s
137	Ubiquitin E3	5	17	4373	496	**8m22s**
138	Ubiquitin E3	5	18	3775	578	4m32s

continua

continuação

Id	Motivo	$\|\mathcal{M}\|$	$\|H^*\|$	#Cortes	#Lazy	Tempo
139	Ubiquitin E3	5	21	4730	1222	6m30s
140	Ubiquitin E3	5	21	3547	772	3m40s

conclusão

Fonte: Ricardo Molinari dos Prazeres, 2022

Apresentamos na Tabela 9 os dados de cada uma das 10 execuções de B&C em cada rede modificada.

Tabela 9 – Ocorrências contidas e não contidas referentes à execução de B&C em diferentes redes modificadas

				Exatas		Aproximadas		
%	Total	Cont	ÑCont	Iguais	Diferentes	ACont	AÑCont	Razão
10	269	227	42	213	21	14	21	0,67
10	275	242	33	229	11	13	22	0,59
10	275	241	34	236	12	5	22	0,23
10	273	230	43	224	10	6	33	0,18
10	276	223	53	213	23	10	30	0,33
10	279	241	38	223	23	18	15	1,20
10	274	243	31	232	19	11	12	0,92
10	276	241	35	228	15	13	20	0,65
10	272	230	42	225	13	5	29	0,17
10	269	244	25	238	9	6	16	0,38
20	270	204	66	183	31	21	35	0,60
20	264	210	54	192	24	18	30	0,60
20	267	208	59	184	18	24	41	0,59
20	271	199	72	174	41	25	31	0,81

continua

continuação

				Exatas		Aproximadas		
%	Total	Cont	ÑCont	Iguais	Diferentes	ACont	AÑCont	Razão
20	258	165	93	152	42	13	51	0,25
20	259	205	54	188	29	17	25	0,68
20	265	183	82	170	47	13	35	0,37
20	270	213	57	186	22	27	35	0,77
20	262	199	63	176	24	23	39	0,59
20	268	216	52	202	30	14	22	0,64
30	253	190	63	160	24	30	39	0,77
30	241	149	92	130	19	19	73	0,26
30	253	192	61	174	26	18	35	0,51
30	257	177	80	150	33	27	47	0,57
30	256	153	103	136	50	17	53	0,32
30	257	184	73	155	22	29	51	0,57
30	246	138	108	118	49	20	59	0,34
30	248	202	46	177	17	25	29	0,86
30	259	172	87	150	16	22	71	0,31
30	263	162	101	146	37	16	64	0,25
40	249	143	106	112	35	31	71	0,44
40	238	146	92	124	44	22	48	0,46
40	248	145	103	129	54	16	49	0,33
40	228	145	83	117	42	28	41	0,68
40	204	121	83	103	36	18	47	0,38
40	253	154	99	132	42	22	57	0,39
40	242	145	97	109	39	36	58	0,62
40	230	147	83	121	25	26	58	0,45
40	245	151	94	122	43	29	51	0,57
40	235	129	106	109	46	20	60	0,33
50	215	119	96	93	17	26	79	0,33
50	224	138	86	113	20	25	66	0,38

continua

continuação

				Exatas		Aproximadas		
%	Total	Cont	ÑCont	Iguais	Diferentes	ACont	AÑCont	Razão
50	222	96	126	66	39	30	87	0,34
50	215	112	103	99	42	13	61	0,21
50	216	120	96	102	37	18	59	0,31
50	213	131	82	111	30	20	52	0,38
50	203	114	89	95	35	19	54	0,35
50	231	114	117	92	44	22	73	0,30
50	215	123	92	95	29	28	63	0,44
50	208	107	101	76	38	31	63	0,49

conclusão

Fonte: Ricardo Molinari dos Prazeres, 2022