



UNIVERSIDADE FEDERAL DO CEARÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE MESTRADO E DOUTORADO EM CIÊNCIA DA
COMPUTAÇÃO - MDCC

MARDSON DA SILVA FERREIRA

PROBLEMA DO ARRANJO LINEAR MÍNIMO

FORTALEZA

2016

Mardson Da Silva Ferreira

Problema do Arranjo Linear Mínimo

Dissertação de Mestrado apresentada ao Programa de Mestrado e Doutorado em Ciência da Computação, do Departamento de Computação da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação. Área de concentração: Otimização (Teoria da Computação).

Universidade Federal do Ceará – UFC

Programa de Mestrado e Doutorado em Ciência da Computação – MDCC

Pró-Reitoria de Pesquisa e Pós-Graduação

Orientador: Prof. Dr. Rafael Castro de Andrade

Fortaleza

6 de setembro de 2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

Ferreira, Mardson da Silva.

Problema do arranjo linear mínimo. / Mardson da Silva Ferreira. – 2016.

69 f. : il.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2016.

Orientação: Prof. Dr. Rafael Castro de Andrade.

1. Otimização combinatória . 2. Programação matemática. 3. Arranjo linear mínimo. I. Título.

CDD 005

MARDSON DA SILVA FERREIRA

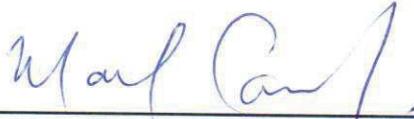
Problema do Arranjo Linear Mínimo

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito para a obtenção do grau de Mestre em Ciência da Computação.

BANCA EXAMINADORA



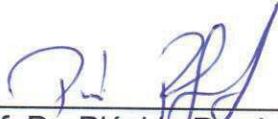
Prof. Dr. Rafael Castro de Andrade
(Presidente)
Universidade Federal do Ceará – UFC



Prof. Dr. Manoel Bezerra Campêlo Neto
Universidade Federal do Ceará – UFC



Prof. Dr. Tibérius de Oliveira e Bonates
Universidade Federal do Ceará – UFC



Prof. Dr. Plácido Rogério Pinheiro
Universidade de Fortaleza – UNIFOR

Fortaleza, 23 de junho de 2016.

Este trabalho é dedicado às pessoas mais importantes da minha vida, minha família.

Agradecimentos

A Deus por sempre ter me dado força para nunca desistir e acreditar que nada é impossível.

Ao meu orientador, Prof. Dr. Rafael Castro de Andrade, por ter aceitado esse desafio, pelo apoio e orientação, sempre presentes e extremamente importantes para a elaboração deste trabalho.

Aos membros da banca avaliadora, pela colaboração na construção desta dissertação.

À minha família, por sempre me apoiar nos momentos mais difíceis, principalmente meus pais, Manoel e Marcia, pela educação e por tudo que fizeram e fazem por mim.

Aos meus amigos do ParGO, que me apoiaram durante todo esse período e contribuíram para este trabalho. Agradeço também pelos momentos de descontração que compartilhamos.

Finalmente à FUNCAP, por permitir a realização deste projeto através do fomento concedido.

*“É muito melhor lançar-se em busca de conquistas grandiosas,
mesmo expondo-se ao fracasso, do que alinhar-se com os pobres de espírito,
que nem gozam muito nem sofrem muito, porque vivem numa penumbra cinzenta,
onde não conhecem nem vitória, nem derrota.
(Theodore Roosevelt)*

Resumo

Seja $G = (V, E)$ um grafo simples e não orientado de conjunto de vértices V e conjunto de arestas E . Dada uma atribuição de rótulos distintos em $\{1, \dots, |V|\}$ aos vértices de G , para cada aresta $uv \in E$, definimos seu peso como sendo a diferença absoluta entre os rótulos atribuídos às suas extremidades. O problema do arranjo linear mínimo (MinLA) é encontrar uma rotulação dos vértices de G de modo que a soma dos pesos de suas arestas seja mínima. MinLA é um problema NP-Difícil cujo poliedro correspondente tem um número fatorial de pontos extremos. Neste trabalho, investigamos um recente modelo quadrático para o MinLA com $\mathcal{O}(|V|^2)$ variáveis e $\mathcal{O}(|V|^2)$ restrições. Esse modelo apresenta o menor número de variáveis e restrições dentre todos os modelos da literatura para o problema. Apresentamos alguns resultados teóricos para o modelo quadrático, bem como mostramos como obter um modelo linear misto cuja solução ótima é a mesma do modelo quadrático. Propomos igualmente desigualdades válidas para os modelos propostos. Apresentamos duas heurísticas para o problema. Implementamos um algoritmo de geração de colunas e um *Branch and Bound* especializado. Experimentos computacionais mostram que o novo modelo teve desempenho superior aos demais modelos conhecidos.

Palavras-chave: Otimização combinatória, Programação matemática, Arranjo linear mínimo.

Abstract

Let $G = (V, E)$ be a simple and undirected graph of set of vertices V and set of edges E . Given an assignment of distinct labels in $\{1, \dots, |V|\}$ to the vertices of G , for every edge $uv \in E$, we define its weight as the absolute difference of labels given to its end nodes. The minimum linear arrangement problem (MinLA) consists in finding a labeling of the vertices of G such that the sum of the weights of its edges is minimized. MinLA is an NP-Hard problem whose corresponding polyhedron has a factorial number of extreme points. In this work, we investigate a recent quadratic model for the MinLA presenting $\mathcal{O}(|V|^2)$ variables and $\mathcal{O}(|V|^2)$ constraints. This model has the smallest number of variables and constraints among all models in the literature for the problem. We present some theoretical results for the quadratic model, and show how to obtain a mixed linear model whose optimal solution is the same of the quadratic one. We also present valid inequalities for the proposed models. We discuss two heuristics for the problem and propose a column generation algorithm and a specialized *Branch and Bound* for MinLA. Computational experiments show that the new quadratic and mixed linear models performed better than existing ones in the literature for new and benchmark instances of this problem.

Keywords: Combinatorial optimization, Mathematical programming, Minimum linear arrangement.

Lista de ilustrações

Figura 1 – Exemplo de um grafo G e rotulações de seus vértices.	26
Figura 2 – Sequências de reduções da demonstração da NP-completude do MinLA.	26
Figura 3 – Exemplo de um ciclo C e um arranjo ótimo com $\varphi(C) = 6$	27
Figura 4 – Ilustração do limite inferior para a soma dos pesos das arestas incidentes em um vértice $u \in V$ de rótulo π_u	29
Figura 5 – Um grafo G , um grafo completo G' e rotulações ótimas dos seus vértices.	30
Figura 6 – Ilustração do caso (I) da inequação (2.16).	37
Figura 7 – Ilustração do caso (II) da inequação (2.16).	38
Figura 8 – Fluxo das ramificações no algoritmo (B_2)	53
Figura 9 – Ilustração de uma ramificação com x 's fracionários. Aqui $\bar{x}_{12} \in (0, 1)$. F_1 e F_2 são os dois nós filhos obtidos fixando-se x_{12} em 0 e em 1, respectivamente.	54
Figura 10 – Ilustração de uma ramificação com π 's inteiros.	55
Figura 11 – Exemplo do uso da estratégia de poda por inviabilidade por conflito de <i>bounds</i>	56

Lista de tabelas

Tabela 1 – Complexidade dos algoritmos de MinLA para certas classes de grafos. . .	27
Tabela 2 – Instâncias da classe I_A	58
Tabela 3 – Instâncias da classe I_B	58
Tabela 4 – Resultados - instâncias da classe I_A	59
Tabela 5 – Resultados - instâncias da classe I_B	60
Tabela 6 – Resultados dos algoritmos de GC_1 e GC_2 - instâncias da classe I_A . . .	61
Tabela 7 – Resultados dos algoritmos de GC_1 e GC_2 - instâncias da classe I_B . . .	61
Tabela 8 – Total de arestas adicionadas pelos algoritmos de geração de colunas - Instâncias classe I_A	62
Tabela 9 – Resultados do modelo (P) - instâncias da classe I_A	62
Tabela 10 – Resultados do modelo (P) - instâncias da classe I_B	63
Tabela 11 – Resultados dos algoritmos de B_1 e B_2 - instâncias da classe I_A	63
Tabela 12 – Novas instâncias aleatórias.	64
Tabela 13 – Resultados do modelo (Q') - instâncias da classe I_A	65
Tabela 14 – Resultados do modelo (Q') - instâncias da classe I_B	65
Tabela 15 – Resultados do modelo (Q') - novas instâncias aleatórias.	65
Tabela 16 – Resultados dos algoritmos de GC_1^* e GC_2^* - instâncias da classe I_A . . .	66
Tabela 17 – Resultados dos algoritmos de GC_1^* e GC_2^* - instâncias da classe I_B . . .	67
Tabela 18 – Resultados do modelo (P') com inclusão de (2.74) em (P) - instâncias da classe I_A	68
Tabela 19 – Resultados do modelo (P') com inclusão de (2.74) em (P) - instâncias da classe I_B	68
Tabela 20 – Resultados dos algoritmos de B_1^* e B_2^* com a inclusão de (2.74) - instân- cias da classe I_A	69
Tabela 21 – Resultados dos algoritmos de B_1^* e B_2^* com a inclusão de 2.74 - instâncias da classe I_B	69
Tabela 22 – Pontuação - instâncias da classe I_A	70
Tabela 23 – Pontuação - instâncias da classe I_B	70
Tabela 24 – Pontuação total de cada um dos modelos para cada critério estudado. .	70
Tabela 25 – Análise via média para as instâncias da classe I_A	71
Tabela 26 – Análise via média para as instâncias da classe I_B	71
Tabela 27 – Resultados do modelo (Q) com <i>Mip Start</i> - instâncias da classe I_A . . .	79
Tabela 28 – Resultados do modelo (Q) com <i>Mip Start</i> - instâncias da classe I_B . . .	80
Tabela 29 – Resultados do modelo (P) com <i>Mip Start</i> - instâncias da classe I_B . . .	80
Tabela 30 – Resultados do modelo (P) com <i>Mip Start</i> - instâncias da classe I_A . . .	81

Tabela 31 – Resultados do modelo (T) - instâncias da classe I_A	84
Tabela 32 – Resultados do modelo (T) - instâncias da classe I_B	84

Índice de algoritmos

1	AUMENTO SUCESSIVO	31
2	CUSTO PARCIAL	31
3	RECOZIMENTO SIMULADO	33

Sumário

	Introdução	23
1	PROBLEMA DO ARRANJO LINEAR MÍNIMO	25
1.1	Definição	25
1.2	Aplicações	25
1.3	Complexidade	26
1.4	Limites para a solução do MinLA	27
1.4.1	Limites inferiores	27
1.4.1.1	Limite inferior obtido a partir de cada vértice do grafo	27
1.4.1.2	Limite inferior obtido do conjunto de arestas	29
1.4.2	Limites superiores obtidos de soluções viáveis	29
2	MODELOS DE PROGRAMAÇÃO MATEMÁTICA	35
2.1	Modelo de Moeini et al.	35
2.1.1	Inequações válidas para o modelo (M') de Moeini et al.	36
2.2	Modelo de Amaral	40
2.3	Modelo de Andrade e Bonates	43
2.3.1	Desigualdades válidas para o modelo (Q)	47
3	ALGORITMOS	51
3.1	Algoritmo de geração de colunas	51
3.2	Branch and Bound	51
3.2.1	Conceitos fundamentais	52
3.2.2	Heurística	52
3.2.3	Estratégias de ramificação	52
3.2.3.1	Ramificação para x 's fracionários	53
3.2.3.2	Ramificação para π_i 's fracionários	53
3.2.3.3	Ramificação para π_i 's inteiros repetidos	54
3.2.4	Técnicas de poda	54
4	RESULTADOS COMPUTACIONAIS	57
4.1	Modelos (Q), (A) e (M')	58
4.2	Algoritmos de geração de colunas	60
4.3	Modelo quadrático (P)	60
4.4	Algoritmos de Branch and Bound	62
4.5	Considerando a desigualdade triangular (2.74) nas variáveis de peso	63

4.5.1	Modelo (Q')	64
4.5.2	Algoritmos de geração de colunas	64
4.5.3	Modelo quadrático (P')	66
4.5.4	Branch and Bound	67
4.6	Analisando os resultados	69
5	CONCLUSÃO E TRABALHOS FUTUROS	73
	Referências	75
	APÊNDICES	77
	APÊNDICE A – RESULTADOS EXTRAS	79
A.1	Resultados do modelo (Q) usando <i>Mip Start</i>	79
A.2	Resultados do modelo (P) usando <i>Mip Start</i>	79
	APÊNDICE B – UM NOVO MODELO TRIVIAL	83
B.1	Resultados do modelo (T)	83

Introdução

Considere $G = (V, E)$ um grafo simples e não orientado de conjunto de vértices V e conjunto de arestas E . Dada uma atribuição de rótulos distintos em $\{1, \dots, |V|\}$ aos vértices de G , para cada aresta $uv \in E$, definimos seu peso como sendo a diferença absoluta entre os rótulos atribuídos às suas extremidades. O problema do arranjo linear mínimo (MinLA) consiste em encontrar uma rotulação dos vértices de G de modo que a soma dos pesos de suas arestas seja mínima.

O problema é NP-Difícil e sua complexidade se mantém mesmo para grafos bipartidos (GAREY; JOHNSON; STOCKMEYER, 1976). Entretanto, para algumas classes de grafos a solução ótima pode ser calculada em tempo polinomial (SCHWARZ, 2010). MinLA tem aplicações em diversas áreas de pesquisa como biologia, matemática e computação (DÍAZ; PETIT; SERNA, 2002).

Existem diversos estudos recentes sobre o problema de MinLA. (SEITZ, 2010) propôs um modelo baseado em distância binária e a partir dele apresenta um interessante estudo poliédrico do problema. (AMARAL, 2009) apresenta um modelo capaz de encontrar soluções ótimas do problema para grafos densos com até 23 vértices. (MOEINI; GUEYE; LOYAL, 2014) propõem um modelo cujo interesse é de encontrar soluções relaxadas de boa qualidade. Uma visão geral de diferentes modelos para o MinLA pode ser encontrada em (SCHWARZ, 2010; SEITZ, 2010). (PETIT, 2003) analisou diferentes técnicas combinatórias para obter limites inferiores e superiores para o problema. Ele defende que as melhores aproximações foram encontradas usando uma meta-heurística conhecida como recozimento simulado.

Neste trabalho, retomamos o estudo do modelo quadrático inicialmente proposto em (ANDRADE; BONATES, 2015). Nossa principal contribuição foi de refinar a nova formulação quadrática, obtendo um modelo compacto misto que apresenta o menor número de variáveis e restrições para o problema. Propomos novas desigualdades válidas e mostramos, com base em resultados numéricos sobre um conjunto de 27 instâncias, que nossa abordagem mostrou ser a mais eficiente em termos computacionais para lidar com esse problema.

O restante do trabalho é organizado como segue. No [Capítulo 1](#) é dada a definição formal do problema, bem como algumas de suas aplicações. Nesse capítulo também falamos sobre sua complexidade e alguns limites inferiores e superiores presentes na literatura. O novo modelo é descrito no [Capítulo 2](#), onde mostramos também outros dois modelos recentes para o problema de MinLA. No [Capítulo 3](#) propomos alguns algoritmos específicos para o modelo de (ANDRADE; BONATES, 2015). O [Capítulo 4](#) contém os resultados das

implementações dos modelos apresentados. Por fim, fazemos algumas considerações finais e discutimos trabalhos futuros no [Capítulo 5](#).

1 Problema do Arranjo Linear Mínimo

Neste capítulo, introduzimos o problema do MinLA e apresentamos sua fundamentação teórica. Apontamos algumas aplicações recorrentes, falamos sobre sua complexidade e apresentamos alguns limites inferiores e heurísticas para o problema.

1.1 Definição

Considere $G = (V, E)$ um grafo simples e não orientado de conjunto de vértices V e conjunto de arestas E . Seja $n = |V|$ e $m = |E|$ denotando o número de vértices e de arestas de G , respectivamente. Sejam $\pi_i \in \{1, \dots, n\}$, para $i = 1, \dots, n$, rótulos todos distintos a serem atribuídos aos n vértices de V . O problema do arranjo linear mínimo consiste em encontrar uma permutação $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ de $\{1, \dots, n\}$ para os vértices de G , de modo a minimizar a expressão

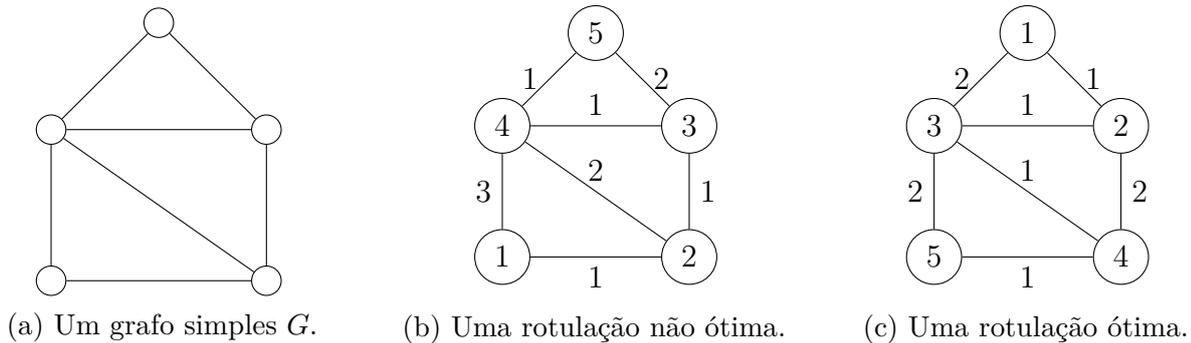
$$\sum_{uv \in E} |\pi_u - \pi_v|. \quad (1.1)$$

A permutação π também é chamada de arranjo ou rotulação. MinLA pertence a uma classe de problemas de otimização combinatória conhecida como problemas de *layout*. Na literatura, esse problema também é conhecido por *Minimum Length Linear Problem*, *Minimum Length Layout Problem*, *Edge Sum Problem* ou *Minimum 1-Sum Problem*. Uma compilação de diversos problemas de *layout* pode ser encontrada em (HARPER, 1964; HANAN; KURTZBERG, 1972; PETIT, 2003).

No exemplo da Figura 1 a seguir, temos um grafo G e duas rotulações para os vértices de G , uma ótima e outra não ótima. Na rotulação ótima da Figura 1(c) a soma dos pesos das arestas de G é a menor possível. Denotamos por $\varphi(G, \pi)$ o valor da rotulação π sobre o grafo G . Usamos $\varphi(G)$ para denotar o valor da rotulação mínima. Observe que na Figura 1(b) temos $\varphi(G, \pi) = 11$, enquanto $\varphi(G) = 10$, como pode ser observado na Figura 1(c). Nas ilustrações apresentadas ao longo do texto, o valor em cada nó é o rótulo do nó e o valor próximo a cada aresta uv é a contribuição $|\pi_u - \pi_v|$ para o valor da função objetivo.

1.2 Aplicações

Um cenário onde o MinLA vem obtendo destaque é no problema de escalonamento de tarefas em uma máquina simples (RAVI; AGRAWAL; KLEIN, 1991; ADOLPHSON, 1977). MinLA pode também ser aplicado no processo de construção de circuitos VLSI

Figura 1 – Exemplo de um grafo G e rotulações de seus vértices.

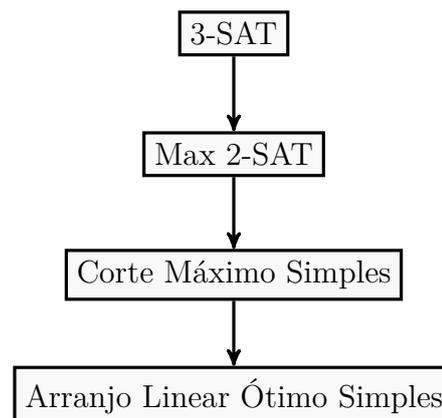
Fonte: Produzido pelos autores.

(*Very-large-scale integration*), na área de desenhos de grafos e em ferramentas de modelagem e diagramação (VANNELLI; ROWAN, 1986; CHEN, 1976; GANE; SARSON, 1977). Para outras aplicações sugerimos (SCHWARZ, 2010; SEITZ, 2010; CAPRARA; LETCHFORD; GONZÁLEZ, 2011).

1.3 Complexidade

O problema de decisão do MinLA, conhecido por Arranjo Linear Ótimo, é NP-completo (GAREY; JOHNSON; STOCKMEYER, 1976). A demonstração da NP-completude do problema foi construída utilizando uma série de reduções a partir do problema de 3-satisfatibilidade booleana, como pode ser observado na Figura 2.

Figura 2 – Sequências de reduções da demonstração da NP-completude do MinLA.



Fonte: Produzido pelos autores.

Apesar do MinLA pertencer à classe dos problemas NP-difíceis (GAREY; JOHNSON; STOCKMEYER, 1976), existem alguns tipos de grafos para os quais o problema pode ser resolvido em tempo polinomial como relatado na Tabela 1. Para grafos como

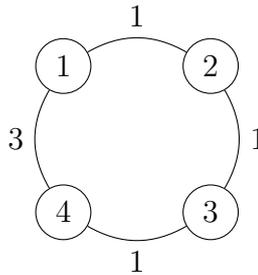
cliques, ciclos e hipercubos o valor é conhecido (SCHWARZ, 2010; PETIT, 2003). Por exemplo, seja C um ciclo com $|V|$ vértices, temos pelo menos n rotulações ótimas que possuem exatamente uma aresta de peso $(n - 1)$ e $(n - 1)$ arestas de peso 1, assim $\varphi(C) = 2(n - 1)$ (veja Figura 3).

Tabela 1 – Complexidade dos algoritmos de MinLA para certas classes de grafos.

Grafo	Complexidade
Malhas quadradas	$O(n)$
Malhas retangulares	$O(n)$
Árvores	$O(n^{\log 3 / \log 2})$
Árvores enraizadas	$O(n \log n)$
Árvores ternárias completas com k níveis	$O(n)$

Fonte: Seitz (2010, p. 32)

Figura 3 – Exemplo de um ciclo C e um arranjo ótimo com $\varphi(C) = 6$.



Fonte: Produzido pelos autores.

1.4 Limites para a solução do MinLA

1.4.1 Limites inferiores

A grande maioria dos limites inferiores para o MinLA ou são combinatórios ou obtidos a partir de programação matemática. Encontrar bons limites inferiores para esse problema não é uma tarefa trivial (AMARAL, 2009). No entanto, existem vários trabalhos que buscam desenvolver métodos para encontrar tais limites. Uma revisão mais aprofundada dos limites inferiores pode ser encontrada em (PETIT, 2001; PETIT, 2003; SEITZ, 2010; SCHWARZ, 2010). Nesta seção, mostramos dois exemplos de limites combinatórios, o primeiro obtido do conjunto de vértices e o segundo do conjunto de arestas do grafo.

1.4.1.1 Limite inferior obtido a partir de cada vértice do grafo

Esse limite inferior foi proposto por (CHUNG, 1988). Ele parte do princípio de que cada vértice u do grafo gera um limite inferior para sua vizinhança. Considere $d(u)$

representando o grau de u e $N(u)$ como sendo o conjunto de vértices adjacentes ao vértice u .

Proposição 1.1. *Seja $w_u = \sum_{v \in N(u)} |\pi_u - \pi_v|$ a soma dos pesos das arestas incidentes em u . Daí, temos que*

- a) $w_u \geq d(u)^2/4 + d(u)/2$, se $d(u)$ for par,
 b) $w_u \geq (d(u)^2 + 2d(u) + 1)/4$, caso contrário.

Demonstração. Seja u um vértice arbitrário de um grafo G e π_u seu rótulo. Observe que, no melhor caso, u tem no máximo dois vizinhos cujos rótulos diferem em uma unidade de π_u , dois vizinhos cujos rótulos diferem em duas unidades de π_u , e assim por diante. Então, temos que u tem no máximo duas arestas com peso 1, duas arestas com peso 2, duas arestas com peso 3, e assim sucessivamente, como ilustrado na [Figura 4](#).

(i) quando $d(u)$ é par, temos que

$$w_u \geq \sum_{i=1}^{d(u)/2} 2i = 2 \sum_{i=1}^{d(u)/2} i = 2 \left[\frac{d(u)}{2} \left(\frac{\frac{d(u)}{2} + 1}{2} \right) \right] = 2 \left(\frac{\frac{d(u)^2}{4} + \frac{d(u)}{2}}{2} \right) \quad (1.2)$$

$$= \frac{d(u)^2}{4} + \frac{d(u)}{2} \quad (1.3)$$

(ii) quando $d(u)$ é ímpar, temos que

$$w_u \geq \frac{1}{2}(d(u) + 1) + \sum_{i=1}^{(d(u)-1)/2} 2i. \quad (1.4)$$

Resolvendo o somatório obtemos

$$\sum_{i=1}^{(d(u)-1)/2} 2i = 2 \left(\sum_{i=1}^{(d(u)-1)/2} i \right) \quad (1.5)$$

$$= 2 \left[\frac{\frac{(d(u)-1)}{2} \left(\frac{(d(u)-1)}{2} + 1 \right)}{2} \right] \quad (1.6)$$

$$= \frac{(d(u) - 1)^2}{4} + \frac{(d(u) - 1)}{2} \quad (1.7)$$

$$= \frac{(d(u)^2 - 1)}{4}. \quad (1.8)$$

Substituindo em (1.4) temos

$$w_u \geq \frac{1}{2}(d(u) + 1) + \frac{(d(u)^2 - 1)}{4} = \frac{d(u)^2 + 2d(u) + 1}{4}. \quad (1.9)$$

□

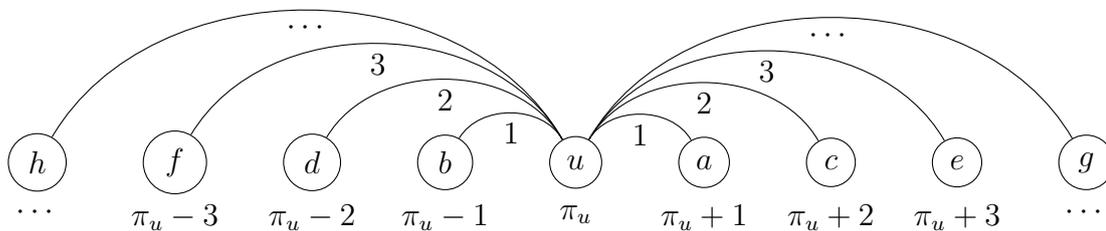
Os casos (i) e (ii) da Proposição 1.1 podem ser expressos pela equação

$$w_u \geq \left\lfloor \frac{(d(u) + 1)^2}{4} \right\rfloor, \forall u \in V. \quad (1.10)$$

Calculando o somatório de w_u , para todo $u \in V$ e dividindo por 2, pois toda aresta uv possui dois vértices adjacentes e assim ela é contada duas vezes, temos o seguinte limite

$$\varphi(G) \geq \frac{1}{2} \sum_{u \in V} w_u. \quad (1.11)$$

Figura 4 – Ilustração do limite inferior para a soma dos pesos das arestas incidentes em um vértice $u \in V$ de rótulo π_u .



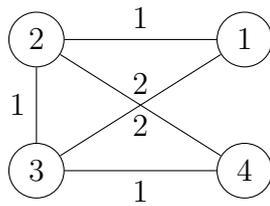
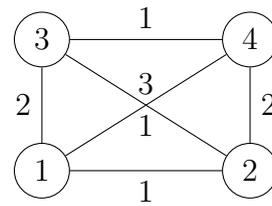
Fonte: Produzido pelos autores

1.4.1.2 Limite inferior obtido do conjunto de arestas

O limite inferior obtido do conjunto de arestas (PETIT, 2001) surgiu da percepção de que, dado um grafo G , com n vértices e uma rotulação π dos vértices de G , existem no máximo $(n - 1)$ arestas com peso 1, $(n - 2)$ arestas com peso 2, e assim sucessivamente. Em geral, existem no máximo $(n - k)$ arestas com peso k em π . Somando os pesos dessas arestas temos um limite inferior para o MinLA. Considere o grafo G da Figura 5(a). Como $n = 4$ e $m = 5$, temos que em uma rotulação ótima para G existem no máximo $(4 - 1)$ arestas com peso 1, $(4 - 2)$ arestas com peso 2. Daí, somando os pesos dessas arestas, totalizamos 7 e obtemos um limite inferior para $\varphi(G)$. Já para o grafo completo G' da Figura 5(b), com $n = 4$ e $m = 6$, temos que existem exatamente $(4 - 1)$ arestas com peso 1, $(4 - 2)$ arestas com peso 2 e $(4 - 3)$ arestas com peso 3. Assim, obtemos o valor 10 como um limite inferior para $\varphi(G')$. Na Figura 5, em ambos os grafos, o limite inferior obtido do conjunto de arestas é igual ao valor da rotulação ótima. No entanto, essa propriedade só é válida para grafos completos.

1.4.2 Limites superiores obtidos de soluções viáveis

Existem diversos métodos heurísticos para o problema do Arranjo Linear Mínimo. Tais métodos utilizam busca local ou procedimentos construtivos, ou ambos. (PETIT, 2001)

Figura 5 – Um grafo G , um grafo completo G' e rotulações ótimas dos seus vértices.(a) grafo G .(b) grafo completo G' .

Fonte: Produzido pelos autores

apresenta uma síntese desses diferentes métodos heurísticos e faz um estudo comparativo entre eles. Dentre os métodos analisados, estão o método construtivo conhecido como *Successive Augmentation* (SCA) e alguns baseados em busca local, como *Hill-Climbing*, *Full-Search*, *Metropolis* e *Simulated Annealing* (SA). (PETIT, 2001) cita que as melhores soluções foram obtidas usando SA, embora tal procedimento exija uma grande quantidade de tempo computacional.

No algoritmo de SCA, uma solução é construída estendendo um arranjo parcial a cada iteração, até que todos os vértices sejam enumerados. O Algoritmo 1 proposto por (PETIT, 2001) funciona como segue. Inicialmente, os vértices são ordenados usando busca em profundidade, pois assim nos beneficiamos da conectividade do grafo. No entanto, outras abordagens poderiam ser utilizadas, como busca em largura ou até mesmo uma ordenação aleatória. Posteriormente, o rótulo “0” é atribuído ao vértice inicial. É como se o vértice inicial fosse colocado no centro de uma reta numérica. A cada iteração, um novo vértice é adicionado ao arranjo, à esquerda ou à direita do vértice inicial, de acordo com o custo parcial relativo à sua adição. As variáveis l e r denotam o número de vértices colocados à esquerda e à direita do vértice inicial, respectivamente. Por fim, é feito um remapeamento dos rótulos atribuídos aos vértices, com base no número de vértices colocados à esquerda. Observe que fazendo $\pi[i] - l$, estamos “empurrando” o rótulo de i de l unidades para à direita.

O Algoritmo 2 retorna o custo do arranjo π restrito aos vértices v_1, \dots, v_{i-1} quando um rótulo k é atribuído ao vértice v_i . O rótulo k representa a próxima posição disponível na reta, à esquerda do vértice inicial, se $k < 0$, ou à direita, caso contrário. A cada iteração é calculada a distância entre k e os rótulos dos vizinhos de v_i na reta numérica. Então, com base na soma dessas distâncias o algoritmo escolhe onde posicionar v_i . Note que, apesar de existirem rótulos com valores positivos e negativos, a variável c que denota o custo parcial é sempre positiva.

O algoritmo de *Metropolis* (METROPOLIS et al., 1953) é uma analogia ao processo térmico de recozimento de metais. O processo consiste em submeter os metais a uma alta

Algoritmo 1: AUMENTO SUCESSIVO**Entrada:** Um grafo $G = (V, E)$ **Saída:** Um layout π

```

1 início
2    $n \leftarrow |V|$ 
3   Selecione uma ordenação inicial dos vértices  $v_1, v_2, \dots, v_n$ 
4    $\pi[v_1] \leftarrow 0$ 
5    $l \leftarrow -1$ 
6    $r \leftarrow 1$ 
7   para  $i \leftarrow 2$  até  $n$  faça
8     se  $CustoParcial(G, \pi, i, v_i, l) < CustoParcial(G, \pi, i, v_i, r)$  então
9        $\pi[v_i] \leftarrow l$  // coloca à esquerda
10       $l \leftarrow l - 1$ 
11     senão
12        $\pi[v_i] \leftarrow r$  // coloca à direita
13        $r \leftarrow r + 1$ 
14     fim
15   fim
16   //remapeando  $\pi$  para  $\{1, \dots, n\}$ 
17   para  $i \leftarrow 1$  até  $n$  faça
18      $\pi[i] \leftarrow \pi[i] - l$ 
19   fim
20 fim
21 retorna  $\pi$ 

```

Algoritmo 2: CUSTO PARCIAL**Entrada:** Um grafo $G = (V, E)$, um layout parcial π , a iteração i corrente, um vértice v_i , um rótulo k **Saída:** custo relativo ao layout parcial π

```

1 início
2    $\pi[v_i] \leftarrow k; c \leftarrow 0$ 
3   para  $j \leftarrow 1$  até  $i$  faça
4     se  $v_i v_j \in E$  então
5        $c \leftarrow c + |\pi[v_i] - \pi[v_j]|$ 
6     fim
7   fim
8 fim
9 retorna  $c$ 

```

temperatura e, em seguida, realizar um resfriamento gradual até que um ponto de congelamento seja alcançado. Nessa analogia, os estados possíveis de um metal correspondem às soluções do espaço de busca. A energia de cada estado equivale ao valor da função objetivo. Esse algoritmo começa gerando uma solução π e uma temperatura t inicial. A cada iteração, uma nova solução $\bar{\pi}$ pertencente à vizinhança de π é selecionada. Se o custo de $\bar{\pi}$ for inferior ao custo de π , então a solução corrente será atualizada, isto é, $\pi \leftarrow \bar{\pi}$.

Caso contrário, será aplicada a fórmula desenvolvida por Boltzmann-Gibbs, dada por $\exp\left(\frac{b-a}{t}\right)$, onde $\exp(x)$ é o valor da função exponencial no ponto x , e b e a são os custos de $\bar{\pi}$ e π , respectivamente. O valor retornado por essa fórmula, digamos ρ , é comparado com $k \in [0, 1)$, gerado aleatoriamente pela função *rand*. Se $\rho > k$, então a nova solução $\bar{\pi}$, mesmo tendo custo superior, será selecionada como solução corrente. A ideia é que, em altas temperaturas, cada solução tem (aproximadamente) a mesma chance de ser a solução corrente. Já em baixas temperaturas, somente soluções em que o valor da função objetivo é pequeno têm alta probabilidade de se tornar a solução corrente.

O algoritmo de *Simulated Annealing* (KIRKPATRICK; VECCHI; GELATT, 1983) consiste de uma sequência de execuções do algoritmo de *Metropolis*, em conjunto com um progressivo decremento da temperatura para gerar soluções de um problema de otimização. No Algoritmo 3 apresentamos uma adaptação do algoritmo SA proposto por (PETIT, 2001). Primeiramente, inicializamos π com uma solução viável, isto é, uma permutação de $\{1, \dots, |V|\}$. Atribuímos um valor alto para a temperatura t inicial. A variável t_{min} representa a condição de parada do algoritmo. O número de soluções geradas para uma determinada temperatura \bar{t} , $t_{min} \leq \bar{t} \leq t$, é definido pela variável eq . Como dito anteriormente, $\varphi(G, \pi)$ é o valor da permutação π sobre o grafo G . Por fim, a variável α define a taxa de decrescimento da temperatura. Na implementação do nosso algoritmo, os parâmetros (t, t_{min}, eq, α) foram inicializados com os mesmos valores usados no algoritmo de (PETIT, 2001).

Algoritmo 3: RECOZIMENTO SIMULADO

Entrada: Um grafo $G = (V, E)$ **Saída:** Um layout π

```

1 início
2    $\pi \leftarrow$  uma solução inicial
3    $t \leftarrow$  temperatura inicial
4    $t_{min} \leftarrow$  ponto de congelamento
5    $eq \leftarrow$  ponto de equilíbrio
6   enquanto  $t > t_{min}$  faça
7      $i \leftarrow 0$ 
8     enquanto  $i \leq eq$  faça
9        $\bar{\pi} \leftarrow$  nova solução
10       $a \leftarrow \varphi(G, \pi)$ 
11       $b \leftarrow \varphi(G, \bar{\pi})$ 
12      se  $b < a$  então
13         $\pi \leftarrow \bar{\pi}$ 
14         $a \leftarrow b$ 
15      senão
16        se  $\exp\left(\frac{b-a}{t}\right) > rand()$  então
17           $\pi \leftarrow \bar{\pi}$ 
18           $a \leftarrow b$ 
19        fim
20      fim
21       $i \leftarrow i + 1$ 
22    fim
23     $t \leftarrow t \cdot \alpha$ , onde  $\alpha \in (0, 1)$ 
24  fim
25 fim
26 retorna  $\pi$ 

```

2 Modelos de programação matemática

Neste capítulo, apresentamos inicialmente dois modelos da literatura para o problema do arranjo linear mínimo; o primeiro, proposto por (MOEINI; GUEYE; LOYAL, 2014), e o segundo, formulado por (AMARAL, 2009). Esses modelos são os mais recentes para o MinLA, além de terem um desempenho satisfatório na resolução do problema. Uma exposição de outros modelos para o MinLA pode ser encontrada em (SEITZ, 2010; SCHWARZ, 2010). Em seu trabalho, (MOEINI; GUEYE; LOYAL, 2014) mostraram que seu modelo consegue obter melhores soluções contínuas em um tempo muito menor que os modelos apresentados em (SEITZ, 2010; SCHWARZ, 2010). Já o modelo de (AMARAL, 2009) encontra soluções ótimas para grafos densos com até 23 vértices.

Em seguida, introduzimos o novo modelo quadrático com $\mathcal{O}(n^2)$ variáveis e $\mathcal{O}(n^2)$ restrições proposto por (ANDRADE; BONATES, 2015), apresentamos resultados teóricos existentes e desenvolvemos novas desigualdades válidas para esse modelo.

2.1 Modelo de Moeini et al.

Uma formulação básica para o MinLA é obtida considerando as seguintes variáveis:

$$x_{ik} = \begin{cases} 1, & \text{se o rótulo } k \text{ é atribuído ao vértice } i, \text{ isto é, } \pi_i = k; \\ 0, & \text{caso contrário,} \end{cases} \quad (2.1)$$

$\forall i, k \in \{1, \dots, n\}$. A formulação é dada a seguir:

$$(M) \quad \min \quad \sum_{ij \in E} \sum_{k=1}^n \sum_{l=1}^n |k - l| x_{ik} x_{jl} \quad (2.2)$$

$$\text{s.a:} \quad \sum_{i=1}^n x_{ik} = 1 \quad \forall k \in \{1, \dots, n\} \quad (2.3)$$

$$\sum_{k=1}^n x_{ik} = 1 \quad \forall i \in \{1, \dots, n\} \quad (2.4)$$

$$x_{ik} \in \{0, 1\} \quad \forall i, k \in \{1, \dots, n\} \quad (2.5)$$

A restrição (2.3) garante que todo rótulo será atribuído a exatamente um vértice, e a restrição (2.4) garante que todo vértice terá exatamente um rótulo. Na função objetivo (2.2) temos que se $\pi_i = k$ e $\pi_j = l$ com $k \neq l$, então o peso da aresta ij é dado pela

diferença absoluta entre k e l . Observe que o modelo (M) não é linear. (MOEINI; GUEYE; LOYAL, 2014) propõem uma linearização de (M) definindo:

$$f_{kl} = \sum_{ij \in E} x_{ik}x_{jl} \in \{0, 1\}, \quad \forall k, l \in \{1, \dots, n\}, \quad k \neq l, \quad (2.6)$$

onde a variável f_{kl} pode ser interpretada como

$$f_{kl} = \begin{cases} 1, & \text{se } \exists e \in E \text{ de extremidades rotuladas com } k \text{ e } l; \\ 0, & \text{caso contrário.} \end{cases} \quad (2.7)$$

Aplicando (2.6) em (2.2), obtemos uma formulação linearizada para o MinLA:

$$(M') \quad \min \quad \sum_{k=1}^n \sum_{l=1}^n |k - l| f_{kl} \quad (2.8)$$

$$\text{s.a:} \quad \sum_{i=1}^n x_{ik} = 1 \quad \forall k \in \{1, \dots, n\} \quad (2.9)$$

$$\sum_{k=1}^n x_{ik} = 1 \quad \forall i \in \{1, \dots, n\} \quad (2.10)$$

$$f_{kl} \geq (x_{ik} + x_{jl} - 1) \quad \forall ij \in E, \quad \forall k, l \in \{1, \dots, n\} \quad (2.11)$$

$$x_{ik} \in \{0, 1\} \quad \forall i, k \in \{1, \dots, n\} \quad (2.12)$$

$$f_{kl} \in \{0, 1\} \quad \forall k, l \in \{1, \dots, n\} \quad (2.13)$$

Infelizmente, a formulação (M') apresenta duas desvantagens. A primeira é o grande número de restrições do tipo (2.11), que é $\mathcal{O}(n^4)$. A segunda é que sua relaxação linear é bastante fraca. Uma solução viável de custo nulo para (M') seria:

$$x_{ik} = \frac{1}{n}, \quad i, k \in \{1, \dots, n\}, \quad (2.14)$$

$$f_{kl} = 0, \quad k, l \in \{1, \dots, n\}. \quad (2.15)$$

2.1.1 Inequações válidas para o modelo (M') de Moeini et al.

Com o propósito de diminuir o número de restrições do tipo (2.11), (MOEINI; GUEYE; LOYAL, 2014), aplicando técnicas de *lifting* (WOLSEY; NEMHAUSER, 1999) em (2.6), propõem o seguinte resultado:

Teorema 2.1 (MOEINI; GUEYE; LOYAL, 2014). *Se definirmos A como sendo a matriz de adjacência de um grafo conexo G , as desigualdades (2.16) e (2.17) a seguir são válidas para o modelo (M') .*

a) para todo $k, l, t \in \{1, \dots, n\}$, $k \neq l$

$$f_{kl} \geq \sum_{j=1}^n A_{tj} x_{jl} + \alpha(1 - x_{tk}), \quad (2.16)$$

em que:

$$\alpha = \min\{A_{i'j'} - A_{tj'} : i', j' \in \{1, \dots, n\} \text{ com } i' \neq t \neq j'\}.$$

b) para todo $k, l, t \in \{1, \dots, n\}$, $k \neq l$

$$f_{kl} \geq \sum_{j=1}^n A_{tj} x_{jl} + \sum_{k'=1, k' \neq k}^n \alpha_{k'} x_{tk'}, \quad (2.17)$$

em que:

$$\alpha_{k'} = \min\{A_{i'j} - A_{tj} : i', j \in \{1, \dots, n\} \text{ com } i' \neq t, j \text{ e } j \neq t \text{ e } k \neq k'\}.$$

Vale ressaltar que o [Teorema 2.1](#) permite substituir as restrições (2.11), que são $\mathcal{O}(n^4)$, por um conjunto de exatamente $n^2(n-1)$ restrições. A demonstração do [Teorema 2.1](#) pode ser encontrada em ([MOEINI; GUEYE; LOYAL, 2014](#)). Para verificar a validade da restrição (2.16) considere os seguintes casos:

- (I) $\exists \bar{j} \in N(t)$ que não é universal¹, isto é, $\alpha = -1$. Se $x_{tk} = 1$, então (2.16) torna-se $f_{kl} \geq \sum_{j=1}^n A_{tj} x_{jl}$ que é válida pela definição de f_{kl} . Nesse caso, se o rótulo l for atribuído a algum vizinho de t ([Figura 6\(a\)](#)), então teremos $f_{kl} \geq 1$; caso contrário, $f_{kl} \geq 0$. Se $x_{tk} = 0$, (2.16) torna-se $f_{kl} \geq \sum_{j=1}^n A_{tj} x_{jl} - 1$ que é válida, pois o somatório é no máximo igual a 1, já que o rótulo l é atribuído a no máximo um vizinho j de t ([Figura 6\(b\)](#));

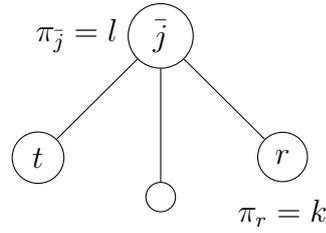
Figura 6 – Ilustração do caso (I) da inequação (2.16).



Fonte: Produzido pelos autores.

- (II) Todo $j \in N(t)$ é universal, isto é, $\alpha = 0$. Desse modo, (2.16) torna-se $f_{kl} \geq \sum_{j=1}^n A_{tj} x_{jl}$. Se o somatório for igual a zero, então a desigualdade é válida de todas as formas. Caso seja 1, existe $\bar{j} \in N(t)$ tal que $x_{\bar{j}l} = 1$. Como \bar{j} é universal,

Figura 7 – Ilustração do caso (II) da inequação (2.16).



Fonte: Produzido pelos autores.

existe vizinho r de \bar{j} com rótulo k . Logo, $f_{kl} = 1$. Como pode ser observado na Figura 7.

A restrição (2.17) é idêntica ao caso (II) de (2.16), quando $x_{tk'} = 0$ para todo $k' \neq k$. No caso complementar, seja $\bar{k}' \neq k$ o rótulo atribuído ao nó t , tal que $x_{t\bar{k}'} = 1$. Procuramos um valor de $\alpha_{\bar{k}'}$ que minimize $f_{kl} - \sum_{j=1}^n A_{tj}x_{jl}$. Temos que podem ocorrer dois casos:

- (i) $\bar{k}' = l$, $\bar{k}' \neq k$ e assim temos que $\alpha_{\bar{k}'} = \min_{j \neq t} A_{tj}$.
- (ii) $\bar{k}' \neq k, l$ e tem-se que $\alpha_{\bar{k}'} = \min_{j \neq t, i' \neq t, i' \neq j} (A_{i'j} - A_{tj})$.

Agora apresentamos as restrições que visam melhorar a relaxação linear de (M') . (MOEINI; GUEYE; LOYAL, 2014) propõem um conjunto de desigualdades válidas que estão relacionadas com a conectividade de G . Observe que $\sum_{l=1}^n f_{kl}$ é igual ao grau do vértice com rótulo k . Sejam $\delta(G)$ e $\Delta(G)$ o grau mínimo e o grau máximo de G , respectivamente. Temos que as desigualdades a seguir são válidas para qualquer grafo conexo G .

- a) Para todo $k \in \{1, \dots, n\}$:

$$\sum_{l=1}^n f_{kl} \geq \delta(G). \quad (2.18)$$

- b) Para todo $k \in \{1, \dots, n\}$:

$$\sum_{l=1}^n f_{kl} \leq \Delta(G). \quad (2.19)$$

- c)

$$\sum_{k=1}^n \sum_{l=1}^n f_{kl} = 2|E|. \quad (2.20)$$

As inequações (2.18) e (2.19) são válidas devido ao fato de existirem pelo menos $\delta(G)$ e no máximo $\Delta(G)$ arestas adjacentes a $v \in V$, respectivamente. Pela fórmula da

¹ Um vértice u é dito universal se para todo $v \in V$ com $u \neq v$, tem-se que $uv \in E$.

soma dos graus, a qual afirma que a soma dos graus dos vértices de G é igual a duas vezes a cardinalidade do conjunto de arestas de G , temos que a equação (2.20) também é válida. Uma versão forte dessas inequações é dada pelo teorema a seguir.

Teorema 2.2 (MOEINI; GUEYE; LOYAL, 2014). *Para todo grafo conexo G , a equação a seguir é válida para (2.8)-(2.13):*

$$\sum_{k=1}^n f_{kl} = \sum_{j=1}^n d(j) \times x_{jl}, \quad l \in \{1, \dots, n\}. \quad (2.21)$$

Demonstração. (MOEINI; GUEYE; LOYAL, 2014). Para todo $l \in \{1, \dots, n\}$, temos que

$$\sum_{k=1}^n f_{kl} = \sum_{k=1}^n \left(\sum_{ij \in E} x_{ik} x_{jl} \right) = \sum_{ij \in E} \left(\sum_{k=1}^n x_{ik} \right) x_{jl}.$$

Como $\sum_{k=1}^n x_{ik} = 1$, tem-se que $\sum_{k=1}^n f_{kl} = \sum_{ij \in E} x_{jl}$. Definindo A como sendo a matriz de adjacência do grafo G temos

$$\sum_{k=1}^n f_{kl} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_{jl} = \sum_{j=1}^n \sum_{i=1}^n A_{ij} x_{jl} = \sum_{j=1}^n x_{jl} \left(\sum_{i=1}^n A_{ij} \right).$$

Uma vez que $\sum_{i=1}^n A_{ij}$ é igual ao grau do vértice j , então a prova está completa. \square

Os seguintes lemas serão úteis para apresentar a próxima desigualdade.

Lema 2.1. $|a + b| \leq |a| + |b|$ para quaisquer que sejam os números reais a e b .

Demonstração. Observe que $|x| \geq x$ e $|x|^2 = x^2$, para todo $x \in \mathbb{R}$. Temos que $|a + b|^2 = (a + b)^2 = a^2 + b^2 + 2ab \leq |a|^2 + |b|^2 + 2|a||b| = (|a| + |b|)^2$. Ou melhor, $|a + b|^2 \leq (|a| + |b|)^2$. Daí obtemos que $|a + b| \leq |a| + |b|$. \square

Lema 2.2. $|a - b| \leq |a - x| + |x - b|$ para quaisquer que sejam os números reais a , b e x .

Demonstração. Note que

$$|a - b| = |(a - x) + (x - b)| \quad \text{aplicando o Lema 2.1} \quad (2.22)$$

$$|a - b| \leq |a - x| + |x - b|. \quad (2.23)$$

\square

Um outro teorema proposto por (MOEINI; GUEYE; LOYAL, 2014) é baseado em desigualdades triangulares. Essas desigualdades estão presentes em diferentes modelos para o MinLA.

Teorema 2.3 (MOEINI; GUEYE; LOYAL, 2014). *As desigualdades a seguir são válidas para (2.8)-(2.13):*

- Se $k, l, m \in \{1, \dots, n\}$ então

$$|k - l|f_{kl} \leq |k - m| + |m - l|, \forall k, l, m, \quad (2.24)$$

$$|k - l|f_{kl} \leq |k - m| + |m - l|f_{ml}, \forall k < l < m, \quad (2.25)$$

- Para todo $k, l \in \{1, \dots, n\}$ e para todo $ij \in E$, se o grau do vértice i for igual a 1, então

$$x_{ik} - x_{jl} + f_{kl} \leq 1. \quad (2.26)$$

Demonstração. (MOEINI; GUEYE; LOYAL, 2014). As desigualdades (2.24) e (2.25) são válidas devido ao fato de que $|k - l| \leq |k - m| + |m - l|$ para toda tripla k, l, m (veja o Lema 2.2) e $f_{kl}, f_{ml} \in \{0, 1\}$ para todo $k, m, l \in \{1, \dots, n\}$. Em (2.26) temos que se $x_{ik} = 1$ e $f_{kl} = 1$ então $x_{jl} = 1$, pois $d(i) = 1$. \square

2.2 Modelo de Amaral

(AMARAL, 2009) propõe um modelo usando o conceito de variáveis de atribuição. Seja γ_{ij} uma variável binária tal que:

$$\gamma_{ij} = \begin{cases} 1, & \text{se o rótulo } j \text{ é atribuído ao vértice } i, \text{ isto é, } \pi_i = j; \\ 0, & \text{caso contrário.} \end{cases} \quad (2.27)$$

Definindo $z_{ipjq} = \gamma_{ip}\gamma_{jq}$ e $u_{ipjqkr} = \gamma_{ip}z_{jqkr}$, $i, j, k \in \{1, \dots, n\}$, $p, q, r \in \{1, \dots, n\}$, produzimos a seguinte formulação de programação não-linear mista 0-1:

$$(A) \quad \min \sum_{\{i,j\} \in E} \sum_{p=1}^n \sum_{q>p}^n \left(z_{ipjq} + \sum_{\substack{k=1 \\ k \neq i \\ k \neq j}}^n \sum_{r=p+1}^{q-1} u_{ipjqkr} \right) \quad (2.28)$$

$$\text{s.a: } \sum_{i=1}^n \gamma_{ij} = 1 \quad 1 \leq j \leq n \quad (2.29)$$

$$\sum_{j=1}^n \gamma_{ij} = 1 \quad 1 \leq i \leq n \quad (2.30)$$

$$\gamma_{ij} \in \{0, 1\} \quad 1 \leq i, j \leq n \quad (2.31)$$

$$z_{ipjq} = \gamma_{ip} \gamma_{jq} \quad 1 \leq i, p, j, q \leq n; i \neq j; p < q \quad (2.32)$$

$$u_{ipjqkr} = \gamma_{ip} z_{jqkr} \quad 1 \leq i, p, j, q, k, r \leq n; p < q < r; j \neq i \neq k \quad (2.33)$$

$$z_{ipjq} \geq 0 \quad 1 \leq i, p, j, q \leq n; q > p; j \neq i \quad (2.34)$$

$$u_{ipjqkr} \geq 0 \quad 1 \leq i, p, j, q, k, r \leq n; p < q < r; i \neq j \neq \quad (2.35)$$

As restrições (2.29) - (2.31) garantem que cada rótulo é atribuído a exatamente um vértice e que cada vértice possui exatamente um rótulo. As restrições (2.32) - (2.33) seguem diretamente das definições de z_{ipjq} e u_{ipjqkr} , respectivamente. Utilizando a técnica de linearização descrita em (ADAMS; SHERALI, 1986), podemos linearizar a restrição (2.32). As restrições a seguir são obtidas multiplicando (2.29) e (2.30) pelas variáveis γ_{ij} apropriadas.

$$\sum_{\substack{j=1 \\ j \neq i}}^n z_{ipjq} = \gamma_{ip}, \quad 1 \leq i, p, q \leq n; p \neq q. \quad (2.36)$$

$$\sum_{\substack{q=1 \\ q \neq p}}^n z_{ipjq} = \gamma_{ip}, \quad 1 \leq i, p, j \leq n; j \neq i. \quad (2.37)$$

As restrições (2.36) e (2.37) em conjunto com (2.29) - (2.31) garantem que as variáveis z_{ipjq} representam o produto $\gamma_{ip} \gamma_{jq}$ (AMARAL, 2009). Para linearizar a restrição (2.33) podemos usar um processo similar. Multiplicando (2.36) e (2.37) pelas variáveis γ_{ij} apropriadas produzimos:

$$\sum_{\substack{i=1 \\ i \neq j \\ i \neq k}}^n u_{ipjqkr} = z_{jqkr}, \quad 1 \leq p, j, q, k, r \leq n; p \neq q \neq r, k \neq j. \quad (2.38)$$

$$\sum_{\substack{p=1 \\ p \neq q \\ p \neq r}}^n u_{ipjqkr} = z_{jqkr} \quad 1 \leq i, j, q, k, r \leq n; k \neq j \neq i, r \neq q. \quad (2.39)$$

(AMARAL, 2009) também propõe algumas desigualdades que permitem diminuir o número de variáveis do modelo. Elas surgem da observação de que $\gamma_{ip}\gamma_{jq} = \gamma_{jq}\gamma_{ip}$ e assim:

$$z_{ipjq} = z_{jqip}, \quad 1 \leq i, j, p, q \leq n; i \neq j; p < q. \quad (2.40)$$

A partir dessa observação, podemos escrever (2.36) como:

$$\sum_{\substack{j=1 \\ j \neq i}}^n z_{ipjq} = \gamma_{ip}, \quad 1 \leq i, p, q \leq n; q > p, \quad (2.41)$$

$$\sum_{\substack{j=1 \\ j \neq i}}^n z_{jqip} = \gamma_{ip}, \quad 1 \leq i, p, q \leq n; q < p, \quad (2.42)$$

e (2.37) como:

$$\sum_{q>p} z_{ipjq} + \sum_{q<p} z_{jqip} = \gamma_{ip}, \quad 1 \leq i, p, j \leq n; j \neq i. \quad (2.43)$$

Uma metodologia análoga pode ser aplicada a (2.38) e (2.39) observando que:

$$u_{ipkrjq} = u_{ipjqkr} = u_{jqipkr} = u_{jqkrip} = u_{krjqip} = u_{kripjq}, \\ 1 \leq i, p, j, q, k, r \leq n; p < q < r; k \neq j \neq i, \quad (2.44)$$

e assim podemos escrever (2.38) e (2.39) como:

$$\sum_{\substack{i=1 \\ i \neq j \\ i \neq k}}^n u_{ipjqkr} = z_{jqkr}, \quad 1 \leq j, p, q, k, r \leq n; p < q < r; k \neq j, \quad (2.45)$$

$$\sum_{\substack{i=1 \\ i \neq j \\ i \neq k}}^n u_{jqipkr} = z_{jqkr}, \quad 1 \leq j, p, q, k, r \leq n; q < p < r; k \neq j, \quad (2.46)$$

$$\sum_{\substack{i=1 \\ i \neq j \\ i \neq k}}^n u_{jqkrip} = z_{jqkr}, \quad 1 \leq j, p, q, k, r \leq n; q < r < p; k \neq j, \quad (2.47)$$

$$\sum_{1 \leq p < q < r \leq n}^n u_{ipjqkr} + \sum_{1 \leq q < p < r \leq n}^n u_{jqipkr} + \sum_{1 \leq q < r < p \leq n}^n u_{jqkrip} = z_{jqkr}, \\ 1 \leq i, j, q, k, r \leq n; r > q; i \neq k \neq j. \quad (2.48)$$

A partir dessas restrições, (AMARAL, 2009) propõe duas formulações para o problema de MinLA:

$$(A_1) \quad \min \quad (2.28)$$

$$\text{s.a.:} \quad (2.29) - (2.31), (2.41) - (2.43), (2.45) - (2.48), (2.34) \text{ e } (2.35)$$

$$(A_2) \quad \min \sum_{p=1}^n \sum_{q>p}^n (q-p) \sum_{\{i,j\} \in E} z_{ipjq}$$

$$\text{s.a.:} \quad (2.29) - (2.31), (2.41) - (2.43), (2.34)$$

Neste trabalho não iremos considerar o modelo (A_1) , pois segundo o próprio autor, o mesmo possui um número muito grande de restrições e variáveis.

2.3 Modelo de Andrade e Bonates

O modelo proposto por (ANDRADE; BONATES, 2015) utiliza a mesma abordagem de ordenação linear das variáveis presente em (SEITZ, 2010). Seja $D = (V, A)$ um grafo direcionado obtido a partir de um grafo simples conexo $G = (V, E)$ com $A = \{(u, v) \mid u, v \in V, u \neq v\}$. Observe que D é um digrafo completo. Considere $A(E) \subset A$ o conjunto de arcos de D onde $(u, v), (v, u) \in A(E) \Leftrightarrow uv \in E$. Seja π uma permutação de $\{1, \dots, |V|\}$ atribuída aos vértices de D . Seja D_π o subdigrafo obtido de D de mesmo conjunto de vértices V e cujos arcos (u, v) pertencem a D_π se e somente se $\pi_v > \pi_u$. Em seu modelo, (ANDRADE; BONATES, 2015) definem variáveis binárias x_{uv} , para todo arco $(u, v) \in A$, em que

$$x_{uv} = \begin{cases} 1 & \text{se } \pi_v > \pi_u, \\ 0, & \text{caso contrário,} \end{cases} \quad (2.49)$$

e w_{uv} para todo arco $(u, v) \in A$, como sendo uma variável usada para estimar o peso de cada arco $(u, v) \in D$. Utilizando tais variáveis, queremos encontrar um subdigrafo D_π cujos arcos são orientados segundo (2.49) e no qual a soma dos w_{uv} tais que $x_{uv} = 1$, com $uv \in E$, seja mínima.

Observação 2.1. *Todo grafo não direcionado $G = (V, E)$, com vértices rotulados por uma função $\ell : V \rightarrow \{1, 2, \dots, |V|\}$, pode ser transformado em um digrafo $D_G = (V, A)$, onde uma aresta $uv \in E$ de G corresponde exatamente a um arco $(u, v) \in A$ se $\ell(v) > \ell(u)$, ou a um arco $(v, u) \in A$, caso contrário.*

O modelo quadrático proposto por (ANDRADE; BONATES, 2015) para o MinLA é dado por:

$$(P) \quad \min_{\pi, x, w} \quad \sum_{(u,v) \in A(E)} w_{uv} x_{uv} \quad (2.50)$$

$$\text{s.a: } x_{uv} + x_{vu} = 1 \quad \forall (u, v) \in A \quad (2.51)$$

$$\pi_v - \pi_u \leq w_{uv} + |V|(1 - x_{uv}) \quad \forall (u, v) \in A \quad (2.52)$$

$$\pi_v - \pi_u \geq w_{uv} - |V|(1 - x_{uv}) \quad \forall (u, v) \in A \quad (2.53)$$

$$1 \leq \pi_v \leq |V| \quad \forall v \in V \quad (2.54)$$

$$1 \leq w_{uv} \leq |V| - 1 \quad \forall (u, v) \in A \quad (2.55)$$

$$x_{uv} \in \{0, 1\} \quad \forall (u, v) \in A \quad (2.56)$$

As restrições (2.51) garantem que exatamente um dos arcos é selecionado entre todo par de vértices u e $v \in V$. As restrições (2.52) e (2.53) estabelecem que se o arco (u, v)

for escolhido, isto é, $x_{uv} = 1$, então $w_{uv} = \pi_v - \pi_u$. Caso contrário, ambas as restrições se tornam redundantes. As demais restrições representam os limites de cada variável. A seguir apresentamos alguns resultados teóricos sobre o modelo (P) .

Teorema 2.4 (ANDRADE; BONATES, 2015). *Dada uma solução viável (π, x, w) de (P) , π é uma permutação de $\{1, 2, \dots, |V|\}$. Além disso, as entradas não-nulas de x induzem um subdigrafo $D_\pi = (V, A_\pi)$ de $D = (V, A)$, com A definido como anteriormente, em que os arcos de A_π são orientados de acordo com a Observação 2.1, sendo os rótulos dos vértices dados pelas variáveis π .*

Demonstração. (ANDRADE; BONATES, 2015). Para todo arco $(u, v) \in A$ tal que $x_{uv} = 1$ temos, pelas restrições (2.52) e (2.53), que $w_{uv} + \pi_u = \pi_v$. Observe que não podemos ter $\pi_u = \pi_v$, pois implicaria $w_{uv} = 0$ com $x_{uv} = 1$, violando assim a restrição (2.55). Assim, para todo arco (u, v) com $x_{uv} = 1$, em qualquer solução viável existe uma diferença de pelo menos uma unidade entre π_u e π_v . Uma vez que D é um digrafo completo e, dados os limites em (2.54), as variáveis π devem ser todas de valores inteiros e correspondem a uma permutação de $\{1, 2, \dots, |V|\}$. Para mostrar que a Observação 2.1 é válida para as entradas não-nulas de x , suponha que existe $u, v \in V$ com $x_{uv} = 1$ e $\pi_u > \pi_v$. Então, (2.52) e (2.53) implicam $\pi_v - \pi_u = w_{uv} < 0$, violando (2.55). \square

Corolário 2.1 (ANDRADE; BONATES, 2015). *Se (π^*, x^*, w^*) é uma solução ótima de (P) , então π^* é uma solução ótima de MinLA.*

Demonstração. (ANDRADE; BONATES, 2015). Pelo Teorema 2.4, π^* é uma permutação de $\{1, 2, \dots, |V|\}$ associada com os vértices em V . Além disso, como D é um digrafo completo, toda permutação de $\{1, 2, \dots, |V|\}$ pode ser convertida em uma solução viável para (P) , o que significa que nenhuma solução para MinLA pode ser perdida por (P) . De modo a estabelecer a otimalidade de π^* , note que a expressão $\sum_{(u,v) \in A(E)} w_{uv}^* x_{uv}^*$ em (2.50) é equivalente a $\sum_{(u,v) \in A(E) | x_{uv}^* = 1} w_{uv}^*$. Assim, o valor z da função objetivo (2.50) correspondendo a x^* é dado por

$$z = \sum_{(u,v) \in A(E) | x_{uv}^* = 1} (\pi_v^* - \pi_u^*) \quad (2.57)$$

$$z = \sum_{(u,v) \in A(E) | x_{uv}^* = 1} |\pi_v^* - \pi_u^*|. \quad (2.58)$$

Onde podemos partir de (2.57) para (2.58) porque a diferença entre os valores de π^* em cada termo é positiva. Portanto, π^* minimizando (2.58) também minimiza (1.1) e a prova está completa. \square

Como descrito em (ANDRADE; BONATES, 2015), podemos linearizar o modelo (P) removendo as variáveis x da função objetivo (2.50) e assim obter o modelo linear inteiro misto (Q).

$$(Q) \quad \min_{\pi, x, w} \sum_{(u,v) \in A(E)} w_{uv} \quad (2.59)$$

s.a: (2.51) – (2.56).

Proposição 2.1 (ANDRADE; BONATES, 2015). *Toda solução ótima de (Q) é também uma solução ótima de (P).*

Demonstração. (ANDRADE; BONATES, 2015). Seja (π^*, x^*, w^*) uma solução ótima de (Q). Nesse caso, π^* é uma permutação ótima de $\{1, 2, \dots, |V|\}$ associada com os vértices em V . Devido ao sentido (min) da otimização, sempre que $x_{uv}^* = 0$ temos que a variável w_{uv}^* correspondente é fixa em seu limite inferior, isto é, em 1. Por outro lado, sempre que $x_{uv}^* = 1$ temos $w_{uv}^* = \pi_v^* - \pi_u^*$, imposto pelo par de restrições ativas (2.52) e (2.53). A contribuição dessas variáveis $w_{uv}^* = 1$ adiciona um valor constante em (2.59). Conseqüentemente, (π^*, w^*, x^*) também é ótima para (2.50). Além disso, se não fosse o caso, uma solução ótima $(\bar{\pi}, \bar{x}, \bar{w})$ de (P) deve apresentar um valor menor que o valor de (π^*, x^*, w^*) em (P). Mas $(\bar{\pi}, \bar{x}, \bar{w})$ também seria viável para (Q), nesse caso teríamos $\bar{x}\bar{w} < x^*w^*$, contradizendo a otimalidade de (π^*, x^*, w^*) para (Q). \square

Corolário 2.2 (ANDRADE; BONATES, 2015). *Se (π^*, x^*, w^*) é uma solução ótima de (Q), então π^* é uma solução ótima de (P) e o valor de sua solução ótima é*

$$\sum_{(u,v) \in A(E)} w_{uv}^* - |E|. \quad (2.60)$$

Demonstração. (ANDRADE; BONATES, 2015). A prova do Corolário 2.2 parte do fato que as variáveis não básicas em uma solução ótima de (Q) estão em seu limite inferior que é um. Observe que $|A(E)| = 2 \cdot |E|$ e por (2.51) somente um arco será escolhido para estar na solução. Assim, temos $|E|$ arcos relacionados a variáveis tais que $x_{uv}^* = 0$ e, conseqüentemente, $w_{uv}^* = 1$ em (Q). Ou seja, o somatório dos pesos dos arcos que não estão na solução é igual a $|E|$. Removendo tal valor daquele em (2.59), obtemos então o valor da solução ótima para (P). \square

O modelo (Q) apresenta um menor número de variáveis e restrições que todos os modelos de programação inteira mista presentes na literatura para esse problema. Mais ainda, somente as variáveis x são restritas a serem inteiras, enquanto as restantes são contínuas. Tanto (P) quanto (Q) podem ser melhorados com o fato de que as restrições (2.52) podem ser substituídas por:

$$\pi_v - \pi_u \leq w_{uv}, \quad \forall (u, v) \in A \quad (2.61)$$

enquanto as restrições (2.55) podem ser fortalecidas quando substituídas por:

$$1 \leq w_{uv} \leq (|V| - 2)x_{uv} + 1, \quad \forall (u, v) \in A \quad (2.62)$$

levando a um novo modelo (Q) dado por:

$$\min_{\pi, x, w} \sum_{(u, v) \in A(E)} w_{uv} \quad (2.63)$$

s.a: (2.51), (2.53), (2.54), (2.56), (2.61), (2.62).

(ANDRADE; BONATES, 2015) mostraram que alguns resultados da literatura se estendem para o novo modelo, como indicado nas proposições seguintes.

Proposição 2.2 (ANDRADE; BONATES, 2015). *Seja $d(v)$ o grau do vértice v em G . Um limite inferior para a soma dos pesos dos arcos de cada vértice v é*

$$\sum_{(u, v) \in A(E)} w_{uv} + \sum_{(v, u) \in A(E)} w_{vu} \geq d(v) + \left\lfloor \frac{(d(v) + 1)^2}{4} \right\rfloor, \quad \forall v \in V. \quad (2.64)$$

Demonstração. (ANDRADE; BONATES, 2015). Note que $2d(v)$ arcos são adjacentes a cada vértice $v \in V$ em D e que em toda solução viável de (Q) nós temos $d(v)$ variáveis w associadas às variáveis x em que $x_{uv} = 0$ e $w_{uv} = 1$. O resultado segue do limite similar mostrado na Subseção 1.4.1.1. \square

Observe que somando os limites acima para todos os vértices $v \in V$ na Proposição 2.2 e dividindo o resultado por dois (cada arco é considerado duas vezes), obtemos um limite inferior válido para a solução de (Q).

Proposição 2.3 (ANDRADE; BONATES, 2015). *Seja p o maior número natural tal que $\sum_{i=1}^p (|V| - i) \leq |E|$ e $\bar{p} = |E| - \sum_{i=1}^p (|V| - i)$. Um limite inferior para o valor da solução ótima z de (Q) é*

$$z \geq |E| + \sum_{i=1}^p (|V| - i)i + \bar{p}(p + 1). \quad (2.65)$$

Demonstração. (ANDRADE; BONATES, 2015). A proposição acima segue do fato de que em toda solução viável de (Q) temos $|E|$ variáveis w correspondentes às componentes nulas de x nos seus limites inferiores e de um resultado equivalente na Subseção 1.4.1.2 para a soma dos pesos das arestas no grafo original G . \square

Proposição 2.4 (ANDRADE; BONATES, 2015). *Uma restrição trivial para (Q) é*

$$\sum_{v \in V} \pi_v = |V|(|V| + 1)/2. \quad (2.66)$$

Proposição 2.5 (ANDRADE; BONATES, 2015). *Um limite válido para π_v , para todo $v \in V$ é*

$$\pi_v \geq 1 + \sum_{(u,v) \in A} x_{uv}. \quad (2.67)$$

Demonstração. (ANDRADE; BONATES, 2015). Segue do fato de que π_v deve ser pelo menos uma unidade maior que o rótulo de qualquer vértice que define um arco apontando para v . Nesse caso, a soma das variáveis que chegam no vértice v é um limite inferior para π_v . \square

2.3.1 Desigualdades válidas para o modelo (Q)

Com o propósito de fortalecer a relaxação do modelo (Q), buscamos traduzir para o contexto desse modelo algumas desigualdades presentes nos modelos de (MOEINI; GUEYE; LOYAL, 2014) e (AMARAL, 2009), bem como introduzir outras novas. Nesta subseção apresentamos algumas desigualdades válidas para os modelos propostos neste trabalho. As desigualdades (2.68) e (2.69) são frequentemente usadas na literatura.

Proposição 2.6. *Seja $T \subseteq D$ tal que T é uma tripla formada pelos vértices u, v e k . As seguintes desigualdades triangulares são válidas para o modelo (Q):*

$$x_{uv} + x_{vk} + x_{ku} \leq 2, \quad \forall \{u, v, k\} \subseteq V, \quad (2.68)$$

$$x_{uk} + x_{kv} + x_{vu} \leq 2, \quad \forall \{k, v, u\} \subseteq V. \quad (2.69)$$

Demonstração. Suponha que $x_{uv} + x_{vk} + x_{ku}$ é estritamente maior que 2, isto é, x_{uv}, x_{vk}, x_{ku} são todas iguais a 1. Pela definição de x temos que $\pi_v > \pi_u$, $\pi_k > \pi_v$ e $\pi_u > \pi_k$, o que é um absurdo. Portanto, tem-se que no máximo duas variáveis de (2.68) podem assumir o valor 1. A prova de (2.69) segue o mesmo princípio. \square

Proposição 2.7. *A igualdade a seguir é válida para (Q):*

$$\sum_{(u,v) \in A | x_{uv}=1} w_{uv} + \sum_{(u,v) \in A | x_{uv}=0} w_{uv} = \frac{|V|(|V| - 1)(|V| + 4)}{6} \quad (2.70)$$

Demonstração. Seja S_1 a soma de w_{uv} quando $x_{uv} = 0$. Observe que em toda solução viável de (Q), temos $|A|/2$ variáveis w com valor 1, isto é, $|V|(|V| - 1)/2$ variáveis w correspondentes às variáveis x nulas. Então, $S_1 = |V|(|V| - 1)/2$. Seja S_2 a soma das variáveis w_{uv} quando $x_{uv} = 1$. Como D é completo e a partir do limite inferior apresentado na Subseção 1.4.1.2, S_2 pode ser expressa como:

$$1 \times (|V| - 1) + 2 \times (|V| - 2) + \dots + (|V| - 1) \times 1.$$

Daí temos que:

$$\begin{aligned}
S_2 &= \sum_{i=1}^{n-1} i(n-i) = \sum_{i=1}^{n-1} i \cdot n - i^2 = n \sum_{i=1}^{n-1} i - \sum_{i=1}^{n-1} i^2 \\
&= n \left[\left(\sum_{i=1}^n i \right) - n \right] - \left[\left(\sum_{i=1}^n i^2 \right) - n^2 \right] \\
&= n \left[\frac{n(n+1)}{2} - n \right] - \left[\frac{n(n+1)(2n+1)}{6} - n^2 \right] \\
&= \left(\frac{n^3 - n^2}{2} \right) - \left(\frac{2n^3 + 3n^2 + n - 6n^2}{6} \right) \\
&= \frac{n^3 - n}{6}.
\end{aligned}$$

Para finalizar a prova, basta somar S_1 e S_2 e assim obter (2.70) em função de n :

$$S_1 + S_2 = \left(\frac{n^2 - n}{2} \right) + \left(\frac{n^3 - n}{6} \right) = \frac{n^3 + 3n^2 - 4n}{6} = \frac{n(n-1)(n+4)}{6}$$

□

Observe que (2.64) age localmente na estrutura de cada vértice, considerando seu grau. Já a restrição (2.70) considera um w global.

Proposição 2.8. *A igualdade abaixo é válida para o modelo (Q):*

$$\pi_u + \sum_{(u,v) \in A} x_{uv} = |V|, \quad \forall u \in V. \quad (2.71)$$

Demonstração. Seja u um vértice qualquer de D_π . Observe que, por (2.49), temos que se $x_{uv} = 1$, então $\pi_v > \pi_u$, $\forall (u,v)$ de D_π . Além disso, $\sum_{(u,v) \in A | x_{uv}=1} x_{uv}$ define o número de arcos que partem de u , tal que $\pi_v > \pi_u$, $\forall v \in V$, isto é, a quantidade de rótulos π_i , $i \in \{1, \dots, |V|\}$ e $i \neq u$, que são estritamente maiores que π_u em D_π . Como toda solução viável de (Q) é uma permutação π de $\{1, \dots, |V|\}$, então a quantidade de arcos partindo de u mais π_u é igual a $|V|$ para todo $u \in V$. □

Corolário 2.3. *A restrição seguinte é válida para o modelo (Q):*

$$\pi_u - \sum_{(v,u) \in A} x_{vu} = 1, \quad \forall u \in V. \quad (2.72)$$

Demonstração. A prova é semelhante à demonstração da Proposição 2.8. Considere u um vértice arbitrário de D_π . Note que para todo arco (v,u) em D_π , $\pi_v < \pi_u$. Além disso, $\sum_{(v,u) \in A | x_{vu}=1} x_{vu}$ corresponde ao número de arcos que chegam em u , tal que $\pi_v < \pi_u$, $\forall v \in V$, isto é, a quantidade de rótulos π_i , $i \in \{1, \dots, |V|\}$ e $i \neq u$, que são estritamente menores que π_u em D_π . Então, π_u menos a quantidade de arcos que chegam em u é igual a 1 para todo $u \in V$. □

Proposição 2.9. Para todo $(u, v) \in A$ temos que

$$w_{uv} + \sum_{(t,u) \in A | t \neq u} x_{tu} \leq |V|. \quad (2.73)$$

Demonstração. Sabemos, pela Proposição 2.5, que o rótulo π_u de um vértice u é pelo menos $1 + \sum_{(t,u) \in A | t \neq u} x_{tu}$. Assim, o peso de todo arco saindo de u seria no máximo $|V| - \left(1 + \sum_{(t,u) \in A | t \neq u} x_{tu}\right) + 1$, onde essa unidade extra refere-se às variáveis w não básicas associadas ao vértice cujo valor do rótulo é $|V|$, sendo que esse limite permanece válido para os demais vértices. \square

Proposição 2.10. Seja uma tripla formada pelos vértices u, v e k . A seguinte desigualdade triangular nos pesos dos arcos induzidos por essa tripla é válida para o modelo (Q) :

$$w_{uv} + w_{vu} \leq w_{vk} + w_{kv} + w_{ku} + w_{uk} - 1, \quad \forall \{u, v, k\} \subseteq V. \quad (2.74)$$

Demonstração. Como dito em (2.51), somente um arco pode ser selecionado entre todo par de vértices. Para todo arco $(u, v) \in A$ tal que $x_{uv} = 1$ temos, pelas restrições (2.52) e (2.53), que $w_{uv} = (\pi_v - \pi_u) = |\pi_v - \pi_u|$, enquanto que $w_{vu} = 1$. Então, temos que para toda tripla $\{u, v, k\} \subseteq V$, exatamente três arcos por ela induzidos terão pesos iguais a 1. E, pelo Lema 2.2, temos que a Proposição (2.10) é válida para (Q) . \square

As Proposições de 2.6 a 2.10 são resultados novos e somam-se aos de (ANDRADE; BONATES, 2015). Doravante, quando falarmos no modelo (Q) , entenda-se o novo modelo (Q) acrescido de (2.68) - (2.73).

3 Algoritmos

Neste capítulo, descrevemos duas versões de um algoritmo de geração de colunas baseado no modelo (Q) da Seção 2.3. Apresentamos também duas versões de um algoritmo de *Branch and Bound* para o problema do Arranjo Linear Mínimo. Primeiro apresentamos as estratégias utilizadas no processo de geração de colunas e posteriormente descrevemos os algoritmos de *B&B*.

3.1 Algoritmo de geração de colunas

Considere o subconjunto $A(E) \subset A$ dos arcos do digrafo D (Seção 2.3). A remoção dos arcos pertencentes a $A - A(E)$ torna D não completo e, como foi visto na prova do Teorema 2.4, D ser completo é uma condição necessária para que as soluções viáveis de (Q) representem uma permutação. Em D não sendo completo, uma solução de (Q) permitiria a existência de π_i 's repetidos.

A ideia é desenvolver um algoritmo de geração de colunas para encontrar a solução ótima para o novo modelo (Q) , considerando inicialmente apenas as restrições restritas ao conjunto de arcos pertencentes a $A(E)$. Se dois vértices u, v da solução ótima parcial tiverem o mesmo rótulo, isto é, $\pi_u = \pi_v$, adicionamos os arcos (u, v) e (v, u) ao conjunto de arcos de D e re-otimizamos o modelo até que a solução ótima encontrada seja uma permutação de $\{1, \dots, |V|\}$, que será ótima para o MinLA. Uma outra versão desse algoritmo, consiste em procurar todos os vértices com rótulos repetidos, adicionar seus respectivos arcos todos de uma vez e só então re-otimizar o modelo.

Nosso objetivo é avaliar qual das situações, partir de um grafo completo ou adicionar arcos iterativamente até a otimalidade, mostra-se mais eficiente e verificar quantos arcos são adicionados pelo algoritmo durante o processo de geração de colunas.

3.2 Branch and Bound

Nas subseções a seguir, explicamos o processo de construção do nosso algoritmo de *B&B*. A Subseção 3.2.1, aborda alguns aspectos gerais do algoritmo. Na Subseção 3.2.2, falamos da heurística utilizada para encontrar um limite superior para o MinLA. Na subseção seguinte, discutimos a estratégia de escolha da variável a ser ramificada, bem como as estratégias de ramificação usadas nos algoritmos. Por fim, a Subseção 3.2.4 descreve as técnicas de poda utilizadas.

3.2.1 Conceitos fundamentais

O algoritmo de *Branch and Bound* é um método de divisão e conquista que utiliza três princípios fundamentais: particionamento (*branching*), avaliação de nós (*bound*) e poda (*prunning*). Associa-se ao *B&B* uma árvore, onde cada nó representa um subproblema obtido do problema original. O particionamento consiste na subdivisão do espaço de soluções viáveis em subespaços menores. Já a poda refere-se à remoção de subproblemas que reconhecidamente não fornecem melhor solução viável que alguma existente para o problema original. Em cada iteração do algoritmo, um nó é escolhido para ser avaliado.

Neste trabalho, o nosso algoritmo de *B&B* avalia os nós usando as relaxações lineares do modelo (Q). Além disso, utilizamos o critério conhecido como *best-bound*, isto é, o nó com menor valor de relaxação é selecionado antes dos demais.

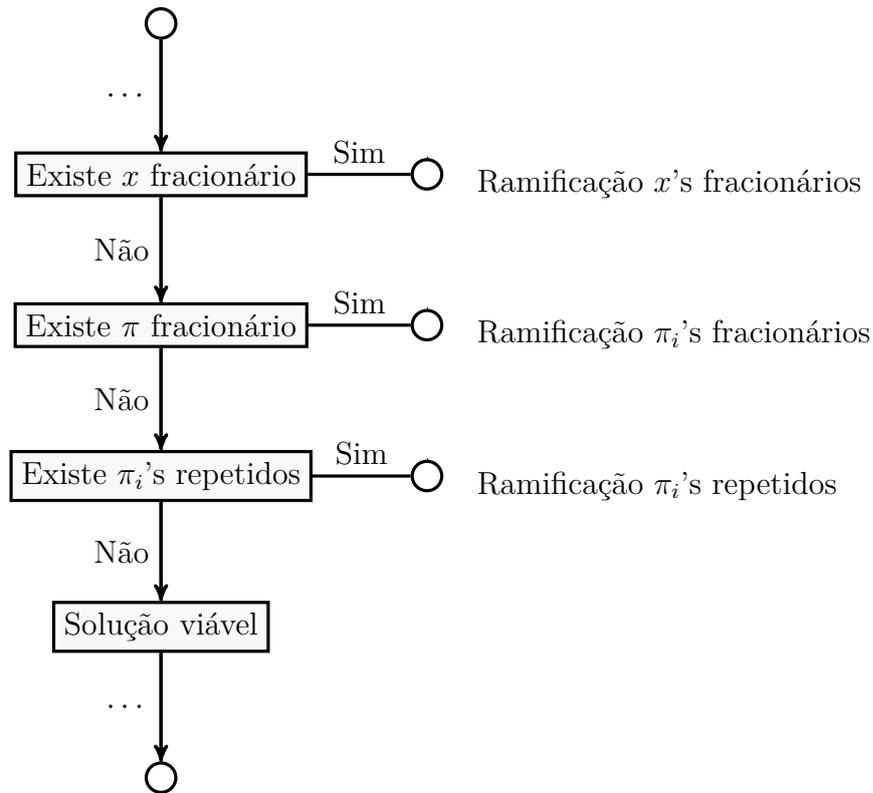
3.2.2 Heurística

Em nosso algoritmo de *B&B* utilizamos a junção dos algoritmos de (SCA) e (SA) mostrados na Subseção 1.4.2. Aplicamos (SCA) dentro do algoritmo de (SA) para encontrar uma solução inicial para o problema. Geramos novas soluções usando uma propriedade específica do poliedro do problema. Podemos, dado uma permutação qualquer, migrar para outra vizinha, trocando de posição dois rótulos cuja diferença absoluta entre eles seja um. Também adotamos a abordagem (SCA + SA) para encontrar uma solução inicial e aplicá-la na estratégia de *MipStart* do CPLEX. Essa estratégia permite atribuir ao CPLEX uma solução inicial. Então o CPLEX utiliza essa solução como ponto de partida no processo de resolução de um problema de programação inteira mista (MIP).

3.2.3 Estratégias de ramificação

Nesta seção, a partir do novo modelo (Q) (veja Seção 2.3), propomos dois algoritmos de *B&B* para o problema de MinLA e apresentamos três tipos de ramificação que são utilizadas nesses algoritmos. No primeiro algoritmo (B_1), utilizamos o digrafo completo D e fazemos uma ramificação binária sobre as variáveis x . No segundo algoritmo (B_2), também usamos o digrafo completo D , mas ramificamos somente as variáveis x relativas aos arcos contidos em $A(E)$. Desse modo, mesmo que todas essas variáveis x sejam inteiras, ainda podemos ter π_i 's fracionários ou rótulos inteiros repetidos. Por isso, empregamos outras duas ramificações, uma para π_i 's fracionários e a outra para π_i 's inteiros repetidos. O fluxo das estratégias de ramificação no algoritmo (B_2) é mostrado na Figura 8.

É importante ressaltar que em ambos, (B_1) e (B_2), cada subproblema é resolvido via programação linear. No processo de escolha da variável a ser ramificada, usamos como critério tomar a variável mais fracionária do nó corrente, isto é, a variável x_{uv} que atinge o valor $\max \min \{ \bar{x}_{uv} - \lfloor \bar{x}_{uv} \rfloor, \lceil \bar{x}_{uv} \rceil - \bar{x}_{uv} \}$, para todo arco (u, v) em A no algoritmo (B_1)

Figura 8 – Fluxo das ramificações no algoritmo (B_2).

Fonte: Produzido pelos autores.

ou $(u, v) \in A(E)$ em (B_2) , onde \bar{x} é a solução corrente. Em caso de empate, escolhamos a primeira variável encontrada entre as variáveis mais fracionárias. Usamos essa mesma metodologia na escolha da variável a ser ramificada referente a π_i 's fracionários.

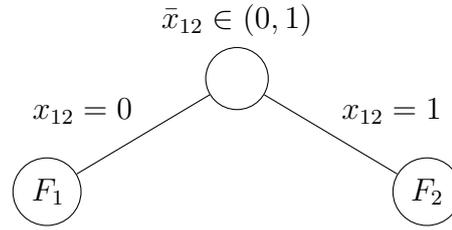
3.2.3.1 Ramificação para x 's fracionários

Esse processo de ramificação é idêntico ao do algoritmo tradicional. Considere \bar{x}_{uv} a variável mais fracionária presente no nó corrente. A partir desse nó criamos dois outros nós na árvore, um com $x_{uv} = 0$ e o outro com $x_{uv} = 1$. Observe que, em ambos os nós, a variável x_{uv} está fixa, ou seja, seu limite inferior e superior são iguais. Na [Figura 9](#), ilustramos um exemplo do processo de ramificação para \bar{x}_{uv} fracionários.

3.2.3.2 Ramificação para π_i 's fracionários

Essa ramificação é semelhante à apresentada anteriormente. No entanto, agora estamos interessados em encontrar o π mais fracionário. Considere $\bar{\pi}_j$ como sendo o valor mais fracionário no nó corrente. Então, geramos dois novos nós, o primeiro com $\pi_j \leq \lfloor \bar{\pi}_j \rfloor$ e o segundo com $\pi_j \geq \lceil \bar{\pi}_j \rceil$.

Figura 9 – Ilustração de uma ramificação com x 's fracionários. Aqui $\bar{x}_{12} \in (0, 1)$. F_1 e F_2 são os dois nós filhos obtidos fixando-se x_{12} em 0 e em 1, respectivamente.



Fonte: Produzido pelos autores.

3.2.3.3 Ramificação para π_i 's inteiros repetidos

Existe um caso particular, no qual, durante o processo de ramificação nas variáveis x no algoritmo B_2 , encontramos nós com π 's inteiros repetidos. Considere um nó arbitrário que obedece essa condição. Seja $F_a(k)$ uma função que verifica se não existe um rótulo fixo com valor k . Em caso positivo, F_a retorna k , caso contrário, retorna o limite anterior a $k - 1$ não fixo. Considere $F_p(k)$ uma função análoga a anterior, mas que procura pelo limite próximo a $k + 1$ não fixo. Escolha um π_j repetido presente no nó sendo avaliado, tal que $\pi_j = \pi_i = k$, onde j representa o vértice do primeiro rótulo repetido e i , o vértice do segundo. Daí, criamos os seguintes nós filhos:

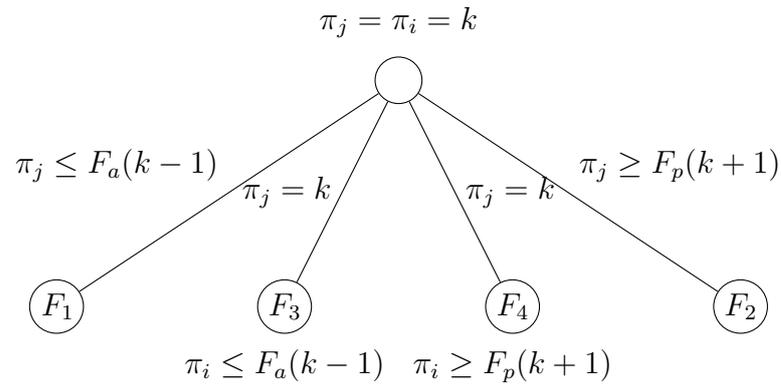
- nó 1: $\pi_j \leq F_a(k - 1)$.
- nó 2: $\pi_j \geq F_p(k + 1)$.
- nó 3: $\pi_i \leq F_a(k - 1)$ e $\pi_j = k$.
- nó 4: $\pi_i \geq F_p(k + 1)$ e $\pi_j = k$.

Esses nós filhos constituem uma partição do espaço de soluções que não contém a solução do nó corrente. Na [Figura 10](#), apresentamos a estrutura dessa ramificação.

3.2.4 Técnicas de poda

No algoritmo de $B\&B$ utilizamos duas estratégias de poda dos nós. A primeira é com base no limite do nó corrente. Dado o limite superior corrente para o problema, podamos os nós cujo valor da solução é maior ou igual a esse limite. Caso a solução obtida no nó seja viável no modelo original e o seu limite menor que o limite superior atual, então o limite superior é atualizado com o valor desse novo limite.

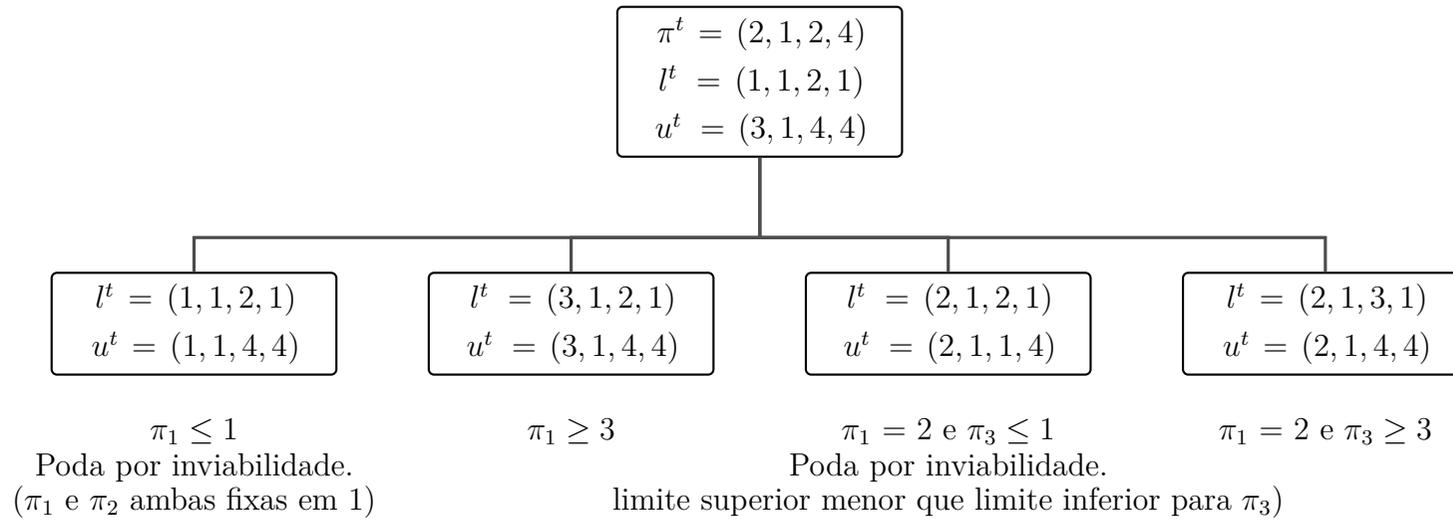
A segunda estratégia baseia-se no princípio de que, como toda solução viável do modelo (Q) é uma permutação, então não podemos ter duas variáveis distintas fixas com o mesmo rótulo. Assim, ao realizarmos a ramificação de uma variável π_i , verificamos se

Figura 10 – Ilustração de uma ramificação com π' s inteiros.

Fonte: Produzido pelos autores.

os limites de π_i em uma das novas partições entra em conflito com os limites de alguma variável π_j , $i \neq j$. Em caso afirmativo o nó deve ser podado, pois as soluções fornecidas pela sua descendência na árvore não seriam uma permutação e, portanto, não seriam uma solução viável para o problema de MinLA. Na [Figura 11](#), os vetores l e u são os limites inferiores e superiores dos valores das variáveis $\pi \in \mathbb{R}_+^4$ de cada nó. O vetor π^t contém os rótulos do nó corrente. Como $\pi_1 = \pi_3$ no nó raiz, então π_1 foi escolhida para ser ramificada. Nesse exemplo, π_2 está fixa.

Figura 11 – Exemplo do uso da estratégia de poda por inviabilidade por conflito de *bounds*.



Fonte: Produzido pelos autores.

4 Resultados Computacionais

Neste capítulo, apresentamos os resultados para os modelos (A), de (AMARAL, 2009), (M'), de (MOEINI; GUEYE; LOYAL, 2014) e o modelo (Q), vistos no Capítulo 2. A implementação dos algoritmos exatos deste trabalho faz uso do pacote de otimização ILOG CPLEX® versão 12.6.1. Neste pacote está presente a biblioteca denominada Concert Technology, que permite o desenvolvimento de modelos matemáticos utilizando a linguagem de programação C++. Os experimentos realizados foram executados utilizando uma máquina com processador Intel Core i7 de oito núcleos de 3.40 GHz, 16.0 GB de memória RAM DDR3 1333 MHz e sistema operacional Linux 14.04 LTS de 64 bits.

As instâncias são divididas em duas classes: I_B , *benchmark* apresentados em (AMARAL, 2009), e I_A , novas instâncias geradas aleatoriamente. As instâncias I_B foram criadas a partir de um conjunto de matrizes propostas por (NUGENT; VOLLMANN; RUML, 1968), disponíveis em (BURKARD; KARISCH; RENDL, 1997), que são construídas como segue: seja Nug a matriz em construção, n sua dimensão e D um *grid* retangular preenchido com números inteiros distintos de 1 até n ; temos que: $Nug[i, j] \leftarrow M(D_i - D_j)$ com $i, j = 1, \dots, n$, onde M corresponde à distância de *Manhattan*¹. As matrizes propostas por (NUGENT; VOLLMANN; RUML, 1968) possuem dimensões 20×20 e 30×30 . As matrizes com dimensões 12, 15, 16 e 17, apresentadas na Tabela 3, foram obtidas removendo-se as últimas linhas e colunas da matriz Nug de dimensão 20. O mesmo processo foi aplicado na matriz Nug de dimensão 30 para obter a de 23.

Uma vez criadas as matrizes de *Nugent*, as instâncias da classe I_B são construídas da seguinte maneira: seja A uma matriz de saída, k um valor obtido de forma aleatória entre os elementos de Nug ; para cada linha i , coluna j da matriz Nug , se $i = j$, $A[i, j] \leftarrow 0$; se $i \neq j$, então: $A[i, j] \leftarrow 0$ se $Nug[i, j]$ for igual a k , e $A[i, j] \leftarrow 1$ caso contrário. Já as instâncias da classe I_A são construídas da seguinte forma: seja A a matriz em construção e $0 < r < 1$ um parâmetro aleatório dado como entrada; inicialmente, A é preenchida aleatoriamente com valores reais entre 0 e 1; para cada linha i em A , $A[i, i] \leftarrow 0$; para cada coluna $j \geq i + 1$ em A , verificamos se $A[i, j]$ é maior que r ; em caso afirmativo, $A[i, j] \leftarrow 1$, caso contrário $A[i, j] \leftarrow 0$, sendo que $A[j, i] \leftarrow A[i, j]$ de modo a obtermos uma matriz simétrica.

As Tabelas 2 e 3 detalham as características das instâncias utilizadas em nossos experimentos. Nelas, n , m , den^2 e z denotam, nessa ordem, o número de vértices, o número de arestas, a densidade e a solução ótima de uma determinada instância. Em

¹ <<http://arxiv.org/abs/1208.5150>>

² A densidade foi calculada usando a fórmula: $den = \frac{2m}{n(n-1)}$.

nossos experimentos, encontramos uma inconsistência no relato dos testes computacionais feito por (AMARAL, 2009). Observamos que o valor da solução ótima das instâncias com 16 e 17 vértices foram reportados incorretamente. Na Tabela 3, esses valores foram corrigidos. As instâncias utilizadas neste trabalho estão disponíveis em <http://www.lia.ufc.br/~mardsonferreira/MinLA/>. Elas foram organizadas em três diretórios: instâncias da classe I_B , instâncias da classe I_A de até 23 nós e as novas instâncias aleatórias com até 26 nós (veja Tabela 12). Em cada instância a primeira linha contém o número de vértices e arestas do grafo, respectivamente. As linhas subsequentes contém os pares (u, v) representando as arestas do grafo.

Tabela 2 – Instâncias da classe I_A .

Instância	n	m	den	z
minla-n10-t0.200-s1	10	34	0,75	100
minla-n10-t0.200-s2	10	36	0,80	108
minla-n10-t0.300-s1	10	33	0,73	100
minla-n10-t0.300-s2	10	34	0,75	98
minla-n10-t0.400-s1	10	25	0,55	64
minla-n10-t0.400-s2	10	31	0,68	84
minla-n10-t0.500-s1	10	25	0,55	64
minla-n10-t0.500-s2	10	22	0,48	54
minla-n10-t0.600-s1	10	15	0,33	30
minla-n10-t0.600-s2	10	20	0,44	43
minla-n10-t0.700-s1	10	14	0,31	27
minla-n10-t0.700-s2	10	14	0,31	29
minla-n10-t0.800-s1	10	10	0,22	17
minla-n10-t0.800-s2	10	12	0,26	21

Fonte: Produzido pelos autores.

Tabela 3 – Instâncias da classe I_B .

Instância	n	m	den	z
GraphNug-n-12-t5	12	61	0,92	241
GraphNug-n-15-t5	15	97	0,92	474
GraphNug-n-16-t6	16	116	0,96	629
GraphNug-n-17-t6	17	131	0,96	748
GraphNug-n-20-t5	20	170	0,89	1.076
GraphNug-n-23-t5	23	221	0,87	1.581

Fonte: Produzido pelos autores.

4.1 Modelos (Q) , (A) e (M')

As Tabelas 4 e 5 detalham os resultados obtidos nos experimentos computacionais. Nelas, $t(s)$ representa o tempo de execução (em segundos) do CPLEX para cada modelo.

As tabelas indicam igualmente o número de iterações ($iter$), que denota o número de iterações *simplex* executadas pelo CPLEX na resolução dos subproblemas contínuos em cada instância e o de nós (bb) avaliados na árvore de B&B do CPLEX.

Analisando os dois conjuntos de instâncias em relação ao tempo de execução, notamos que o modelo (Q) foi melhor que o modelo (A) em todas as instâncias. Em apenas uma instância, GraphNug-n-16-t6, o tempo de execução do modelo (M') foi menor, em 1s, que o do modelo (Q) . E somente em uma das instâncias, GraphNug-n-17-t6, os tempos de execução de ambos os modelos (M') e (Q) foram equivalentes. Quanto ao número de subproblemas (bb) resolvidos pelo B&B do CPLEX, em todas as instâncias o número de nós resolvidos pelo modelo (A) foi menor que o do modelo (Q) . Em 18 instâncias o número de nós resolvidos pelo modelo (Q) foi menor que o do modelo (M') . Em apenas 2 instâncias o número de nós resolvidos pelo modelo (M') foi menor que o do modelo (Q) . Já em termos de número de iterações ($iter$), em 7 instâncias o número de iterações do modelo (Q) foi menor que o do modelo (A) . Nas outras 13 instâncias o número de iterações do modelo (A) foi menor que o do modelo (Q) . Em 17 instâncias o número de iterações do modelo (Q) foi menor que o do modelo (M') . Em somente 3 instâncias o número de iterações do modelo (M') foi menor que o do modelo (Q) .

No que se refere ao tempo de execução, o modelo (Q) se mostrou mais eficiente que os demais. Já nos quesitos número de iterações e nós resolvidos, o modelo (A) foi quem obteve melhor desempenho, seguido de perto pelo modelo (Q) . Realizamos alguns experimentos com o modelo (Q) em conjunto com a estratégia de *Mip Start*, porém como os resultados não foram muito relevantes optamos por colocá-los no apêndice A.

Tabela 4 – Resultados - instâncias da classe I_A .

Instância	(A)			(M')			(Q)		
	bb	$iter$	$t(s)$	bb	$iter$	$t(s)$	bb	$iter$	$t(s)$
minla-n10-t0.200-s1	0	13.911	7	6.564	254.884	8	233	15.049	1
minla-n10-t0.200-s2	0	10.066	3	1.485	48.310	2	94	8.371	0
minla-n10-t0.300-s1	0	24.632	9	11.428	430.498	11	701	28.663	0
minla-n10-t0.300-s2	0	1.137	2	3.039	96.913	3	120	9.543	0
minla-n10-t0.400-s1	29	107.932	38	12.782	591.228	13	4.050	123.793	2
minla-n10-t0.400-s2	0	16.792	9	1.621	59.449	3	467	24.577	0
minla-n10-t0.500-s1	13	91.482	24	16.543	771.751	16	4.332	168.918	1
minla-n10-t0.500-s2	21	100.728	28	13.420	694.767	14	4.661	126.023	1
minla-n10-t0.600-s1	24	155.420	40	10.507	427.863	10	1.479	58.784	1
minla-n10-t0.600-s2	0	43.585	15	10.986	505.198	12	2.012	80.289	1
minla-n10-t0.700-s1	23	83.820	23	7.866	420.430	8	1.838	87.498	1
minla-n10-t0.700-s2	23	137.612	36	15.841	1.032.202	18	2.708	94.695	1
minla-n10-t0.800-s1	0	51.985	20	1.816	92.542	2	1.268	76.638	1
minla-n10-t0.800-s2	9	23.718	8	2.217	124.721	3	876	50.854	0

Fonte: Produzido pelos autores.

Tabela 5 – Resultados - instâncias da classe I_B .

Instância	(A)			(M')			(Q)		
	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)
GraphNug-n-12-t5	0	8	4	57	4.240	1	28	4.620	0
GraphNug-n-15-t5	0	9	15	1.619	197.261	18	836	170.749	3
GraphNug-n-16-t6	6	366.065	937	150	5.554	2	639	201.245	3
GraphNug-n-17-t6	5	321.741	1.300	262	16.881	5	574	190.723	5
GraphNug-n-20-t5	8	4.733.536	84.120	107.190	10.825.888	5.826	10.143	3.343.838	139
GraphNug-n-23-t5	3	452.931	8.929	*	*	*	321.554	95.474.437	5.981

* Execução abortada por falta de memória no PC.

Fonte: Produzido pelos autores.

4.2 Algoritmos de geração de colunas

Agora apresentamos os resultados para os algoritmos de geração de colunas vistos na Seção 3.1. No primeiro, a cada iteração, adicionamos um arco cujas extremidades tem rótulos repetidos e re-otimizamos o modelo. Já no segundo, adicionamos em cada nó todos os arcos que apresentam rótulos repetidos. Denominamos esses algoritmos por GC_1 e GC_2 , respectivamente.

Como pode ser observado nas Tabelas 6 e 7, em todas as instâncias o número de iterações do modelo (Q) foi menor que o do algoritmo GC_1 , e em 13 instâncias foi menor que o do algoritmo GC_2 . Em apenas 3 instâncias o número de iterações do algoritmo GC_2 foi menor do que o do modelo (Q). Com relação ao número de subproblemas resolvidos pelo B&B do CPLEX, em todas as instâncias o número de nós resolvidos pelo modelo (Q) foi menor que o dos algoritmos GC_1 e GC_2 . Em todas as instâncias o tempo de execução do modelo (Q) foi melhor que o do algoritmo GC_1 . Em 12 instâncias o modelo (Q) obteve tempo de execução inferior ao do algoritmo GC_2 . Houve empate em apenas 3 instâncias. É importante ressaltar que em 5 instâncias do grupo I_B , ambos os algoritmos GC_1 e GC_2 , ultrapassaram o limite de tempo de 5 horas e tiveram sua execução abortada. Os resultados mostram que o modelo (Q) foi melhor que os algoritmos GC_1 e GC_2 em todos os critérios analisados.

Outro ponto que verificamos nesses algoritmos foi a quantidade de arestas adicionadas até alcançar o ótimo. A Tabela 8 contém os resultados para as instâncias da classe I_A . Os resultados para as instâncias da classe I_B foram omitidos, pois em praticamente todas as instâncias dessa classe, tanto GC_1 quanto GC_2 ultrapassaram o limite de tempo estipulado. Observamos que mesmo adicionando poucas arestas, essa abordagem se mostrou ineficiente.

4.3 Modelo quadrático (P)

Também realizamos alguns testes resolvendo diretamente o modelo quadrático (P) com o CPLEX. Os resultados estão nas Tabelas 9 e 10. Analisando ambos os conjuntos de

Tabela 6 – Resultados dos algoritmos de GC_1 e GC_2 - instâncias da classe I_A .

Instância	GC_1			GC_2			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
minla-n10-t0.200-s1	22.685	573.935	5	22.685	573.935	5	233	15.049	1
minla-n10-t0.200-s2	28.974	684.982	7	28.974	684.982	7	94	8.371	0
minla-n10-t0.300-s1	31.278	540.989	14	24.879	518.431	15	701	28.663	0
minla-n10-t0.300-s2	19.697	533.224	10	18.841	500.188	11	120	9.543	0
minla-n10-t0.400-s1	6.999	148.224	4	6.989	197.954	2	4.050	123.793	2
minla-n10-t0.400-s2	14.601	428.629	9	14.861	426.104	15	467	24.577	0
minla-n10-t0.500-s1	6.349	169.682	5	10.138	253.251	8	4.332	168.918	1
minla-n10-t0.500-s2	6.284	131.234	4	7.641	164.224	3	4.661	126.023	1
minla-n10-t0.600-s1	3.058	86.775	3	2.777	93.025	1	1.479	58.784	1
minla-n10-t0.600-s2	2.816	110.062	2	3.089	110.797	1	2.012	80.289	1
minla-n10-t0.700-s1	2.890	104.533	5	2.676	69.467	2	1.838	87.498	1
minla-n10-t0.700-s2	4.444	108.076	6	5.250	107.326	6	2.708	94.695	1
minla-n10-t0.800-s1	1.568	44.532	2	2.521	68.156	2	1.268	76.638	1
minla-n10-t0.800-s2	1.303	41.679	2	1.311	57.500	1	876	50.854	0

Fonte: Produzido pelos autores.

Tabela 7 – Resultados dos algoritmos de GC_1 e GC_2 - instâncias da classe I_B .

Instância	GC_1			GC_2			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
GraphNug-n-12-t5	2.284.128	59.832.230	1.132	2.284.128	59.832.230	946	28	4.620	0
GraphNug-n-15-t5	*	*	*	*	*	*	836	170.749	3
GraphNug-n-16-t6	*	*	*	*	*	*	639	201.245	3
GraphNug-n-17-t6	*	*	*	*	*	*	574	190.723	5
GraphNug-n-20-t5	*	*	*	*	*	*	10.143	3.343.838	139
GraphNug-n-23-t5	*	*	*	*	*	*	321.554	95.474.437	5.981

* Execução abortada depois que o limite de 5 horas foi alcançado.

Fonte: Produzido pelos autores.

instâncias em relação ao tempo de execução, observamos que em 10 instâncias o modelo (P) obteve melhor desempenho que o modelo (Q). Em 4 instâncias o modelo (Q) obteve tempo de execução menor que o do modelo (P). Nas outras 6 instâncias os tempos de execução de ambos os modelos (P) e (Q) foram iguais. Vale ressaltar que em apenas duas instâncias da classe I_B foi possível notar uma diferença significativa de tempo. Quanto ao número de subproblemas resolvidos pelo B&B do CPLEX, em 19 instâncias o número de nós resolvidos pelo modelo (P) foi menor que o do modelo (Q). Em apenas 1 instância, minla-n10-t0.400-s2, o número de nós resolvidos pelo modelo (Q) foi menor que o do modelo (P). Já em termos de número de iterações, em 19 instâncias o número de iterações do modelo (P) foi menor que o do modelo (Q). Em apenas 1 instância, minla-n10-t0.400-s2, o número de iterações do modelo (Q) foi menor que o do modelo (P). Portanto, temos que o modelo (P) obteve um desempenho melhor que o modelo (Q) em todos os parâmetros analisados.

Assim como fizemos no modelo (Q), também aplicamos a estratégia de *Mip Start* em (P). E, similarmente ao que aconteceu no modelo (Q), os resultados não foram expressivos e portanto preferimos colocá-los no apêndice A.

Tabela 8 – Total de arestas adicionadas pelos algoritmos de geração de colunas - Instâncias classe I_A .

Instância	Total de arestas adicionadas em GC ₁	Total de arestas adicionadas em GC ₂
minla-n10-t0.200-s1	1	1
minla-n10-t0.200-s2	1	1
minla-n10-t0.300-s1	2	4
minla-n10-t0.300-s2	1	2
minla-n10-t0.400-s1	3	1
minla-n10-t0.400-s2	1	4
minla-n10-t0.500-s1	3	6
minla-n10-t0.500-s2	3	2
minla-n10-t0.600-s1	3	2
minla-n10-t0.600-s2	2	2
minla-n10-t0.700-s1	7	6
minla-n10-t0.700-s2	9	14
minla-n10-t0.800-s1	4	6
minla-n10-t0.800-s2	3	2

Fonte: Produzido pelos autores.

Tabela 9 – Resultados do modelo (P) - instâncias da classe I_A .

Instância	(P)			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
minla-n10-t0.200-s1	0	632	0	233	15.049	1
minla-n10-t0.200-s2	0	912	1	94	8.371	0
minla-n10-t0.300-s1	596	27.519	1	701	28.663	0
minla-n10-t0.300-s2	106	8.719	0	120	9.543	0
minla-n10-t0.400-s1	3.327	106.295	1	4.050	123.793	2
minla-n10-t0.400-s2	945	40.940	1	467	24.577	0
minla-n10-t0.500-s1	2.762	61.675	1	4.332	168.918	1
minla-n10-t0.500-s2	1.701	42.607	1	4.661	126.023	1
minla-n10-t0.600-s1	926	22.580	0	1.479	58.784	1
minla-n10-t0.600-s2	1.564	40.063	1	2.012	80.289	1
minla-n10-t0.700-s1	790	20.183	1	1.838	87.498	1
minla-n10-t0.700-s2	1.795	40.385	0	2.708	94.695	1
minla-n10-t0.800-s1	408	13.900	0	1.268	76.638	1
minla-n10-t0.800-s2	411	22.435	1	876	50.854	0

Fonte: Produzido pelos autores.

4.4 Algoritmos de Branch and Bound

Agora apresentamos os resultados para os algoritmos de B&B mostrados na Seção 3.2. Tanto o algoritmo B₁ quanto B₂ excederam o tempo limite de 5 horas para todas as instâncias da classe I_B . Então apresentamos somente os resultados para as instâncias da classe I_A .

Em todas as instâncias o número de subproblemas resolvidos pelo B&B do CPLEX no modelo (Q) foi menor que o do algoritmo B₁, e em 10 instâncias foi menor que o do algoritmo B₂. Em apenas 4 instâncias o número nós resolvidos pelo algoritmo B₂ foi menor

Tabela 10 – Resultados do modelo (P) - instâncias da classe I_B .

Instância	(P)			(Q)		
	bb	$iter$	$t(s)$	bb	$iter$	$t(s)$
GraphNug-n-12-t5	0	1.115	0	28	4.620	0
GraphNug-n-15-t5	0	2.308	1	836	170.749	3
GraphNug-n-16-t6	0	3.264	1	639	201.245	3
GraphNug-n-17-t6	0	3.195	1	574	190.723	5
GraphNug-n-20-t5	6.672	1.297.321	52	10.143	3.343.838	139
GraphNug-n-23-t5	181.153	62.023.724	5.057	321.554	95.474.437	5.981

Fonte: Produzido pelos autores.

que o do modelo (Q) . Quanto ao tempo de execução, em todas as instâncias o tempo de execução do modelo (Q) foi melhor que o do algoritmo B_1 . Em 13 instâncias o modelo (Q) obteve tempo de execução inferior ao do algoritmo B_2 . E em apenas uma instância, minla-n10-t0.800-s1, o algoritmo B_2 foi melhor que o modelo (Q) . Desse modo, o modelo (Q) conseguiu melhores resultados que os algoritmos B_1 e B_2 .

Tabela 11 – Resultados dos algoritmos de B_1 e B_2 - instâncias da classe I_A .

Instância	B_1		B_2		(Q)	
	bb	$t(s)$	bb	$t(s)$	bb	$t(s)$
minla-n10-t0.200-s1	11.478	51	28.374	111	233	1
minla-n10-t0.200-s2	7.406	33	28.374	112	94	0
minla-n10-t0.300-s1	20.798	95	43.088	168	701	0
minla-n10-t0.300-s2	4.178	19	21.440	89	120	0
minla-n10-t0.400-s1	18.018	70	8.084	30	4.050	2
minla-n10-t0.400-s2	7.168	30	14.582	55	467	0
minla-n10-t0.500-s1	16.010	62	8.244	30	4.332	1
minla-n10-t0.500-s2	14.512	55	6.858	24	4.661	1
minla-n10-t0.600-s1	5.742	20	1.972	7	1.479	1
minla-n10-t0.600-s2	7.128	27	2.908	10	2.012	1
minla-n10-t0.700-s1	4.174	14	1.306	4	1.838	1
minla-n10-t0.700-s2	11.292	39	1.922	6	2.708	1
minla-n10-t0.800-s1	3.694	13	266	0	1.268	1
minla-n10-t0.800-s2	4.862	16	461	1	876	0

Fonte: Produzido pelos autores.

4.5 Considerando a desigualdade triangular (2.74) nas variáveis de peso

Seguem os resultados dos algoritmos anteriores com a inclusão da desigualdade (2.74) mostrada na Subseção 2.3.1. Denominamos de (Q') e (P') os modelos (Q) e (P) adicionados dessa restrição. Como veremos a seguir, essa inclusão tem um impacto significativo nos algoritmos e permite ao modelo (Q') resolver instâncias maiores. Por isso criamos um outro conjunto de instâncias aleatórias com 24, 25 e 26 vértices. A Tabela 12 detalha

as características das instâncias utilizadas nos experimentos. Nas seções subsequentes comparamos os novos resultados com os do modelo (Q) apresentados na Seção 4.1.

Tabela 12 – Novas instâncias aleatórias.

Instância	n	m	den	z
minla-n24-t0.100	24	246	0,89	1.890
minla-n24-t0.200	24	233	0,84	1.728
minla-n24-t0.300	24	201	0,72	1.368
minla-n25-t0.100	25	271	0,90	2.147
minla-n25-t0.200	25	235	0,78	1.746
minla-n26-t0.100	26	289	0,88	2.732
minla-n26-t0.200	26	263	0,80	2.043

Fonte: Produzido pelos autores.

4.5.1 Modelo (Q')

As Tabelas 13 e 14 contêm os resultados dos experimentos para as instâncias usadas na Seção 4.1. Nelas, podemos ver que, com relação ao tempo de execução, em 6 instâncias o modelo (Q') foi melhor que o modelo (Q) . Em 6 instâncias o modelo (Q) obteve tempo de execução menor que o do modelo (Q') . Nas outras 8 instâncias os tempos de execução de ambos os modelos (Q') e (Q) foram iguais. Quanto ao número de subproblemas resolvidos pelo B&B do CPLEX, em 15 instâncias o número de nós resolvidos pelo modelo (Q') foi menor que o do modelo (Q) . Em 5 instâncias o número de nós resolvidos pelo modelo (Q) foi menor que o do modelo (Q') . Já em termos de número de iterações, em 18 instâncias o número de iterações do modelo (Q') foi menor que o do modelo (Q) . Em apenas 2 instâncias o número de iterações do modelo (Q) foi menor que o do modelo (Q') . Ambos os modelos tiveram desempenho semelhante com relação ao tempo de execução. Já nos quesitos número de nós resolvidos e iterações, o modelo (Q') teve um desempenho melhor que o modelo (Q) .

Os resultados para o novo conjunto de instâncias é apresentado na Tabela 15 a seguir. Nela, podemos observar que o modelo (Q') se mostra eficiente para grafos densos com até 26 vértices. Como o modelo (Q) só encontra soluções para grafos com até 23 vértices, então não foi possível fazer um comparativo entre os modelos.

4.5.2 Algoritmos de geração de colunas

Refizemos os experimentos dos algoritmos GC_1 e GC_2 considerando as desigualdades (2.74) em (Q) . Denominamos esses algoritmos como GC_1^* e GC_2^* , respectivamente. Como pode ser observado nas Tabelas 16 e 17, em 16 instâncias o número de iterações do modelo (Q) foi menor que o do algoritmo GC_1^* , e em 4 instâncias o número de iterações do algoritmo

Tabela 13 – Resultados do modelo (Q') - instâncias da classe I_A .

Instância	(Q')			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
minla-n10-t0.200-s1	0	420	1	233	15.049	1
minla-n10-t0.200-s2	0	403	1	94	8.371	0
minla-n10-t0.300-s1	694	19.625	1	701	28.663	0
minla-n10-t0.300-s2	0	411	0	120	9.543	0
minla-n10-t0.400-s1	436	18.033	1	4.050	123.793	2
minla-n10-t0.400-s2	0	420	1	467	24.577	0
minla-n10-t0.500-s1	383	14.101	1	4.332	168.918	1
minla-n10-t0.500-s2	646	24.364	0	4.661	126.023	1
minla-n10-t0.600-s1	404	13.857	0	1.479	58.784	1
minla-n10-t0.600-s2	301	9.374	0	2.012	80.289	1
minla-n10-t0.700-s1	1.073	30.064	1	1.838	87.498	1
minla-n10-t0.700-s2	5.067	150.993	1	2.708	94.695	1
minla-n10-t0.800-s1	1.286	39.166	1	1.268	76.638	1
minla-n10-t0.800-s2	915	28.758	1	876	50.854	0

Fonte: Produzido pelos autores.

Tabela 14 – Resultados do modelo (Q') - instâncias da classe I_B .

Instância	(Q')			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
GraphNug-n-12-t5	0	707	0	28	4.620	0
GraphNug-n-15-t5	582	25.361	3	836	170.749	3
GraphNug-n-16-t6	1.327	183.700	10	639	201.245	3
GraphNug-n-17-t6	718	318.165	17	574	190.723	5
GraphNug-n-20-t5	7.907	770.576	105	10.143	3.343.838	139
GraphNug-n-23-t5	6.378	2.965.477	646	321.554	95.474.437	5.981

Fonte: Produzido pelos autores.

Tabela 15 – Resultados do modelo (Q') - novas instâncias aleatórias.

Instância	(Q')		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
minla-n24-t0.100	13.284.995	11.148	2.575
minla-n24-t0.200	11.096.338	23.080	2.030
minla-n24-t0.300	8.458.689	6.743	1.807
minla-n25-t0.100	19.445.308	5.347	702
minla-n25-t0.200	23.415.045	11.845	5.088
minla-n26-t0.100	6.216.631	8.825	2.336
minla-n26-t0.200	32.083.070	18.938	10.824

Fonte: Produzido pelos autores.

GC_1^* foi menor. Já o algoritmo GC_2^* obteve um número de iterações menor que (Q) em apenas 2 instâncias e nas outras 18 o modelo (Q) teve um número menor de iterações. Com relação ao número de subproblemas resolvidos pelo B&B do CPLEX, em 16 instâncias o número de nós resolvidos pelo modelo (Q) foi menor que o dos algoritmos GC_1^* e GC_2^* . E em 4 instâncias, ambos os algoritmos resolveram menos subproblemas que o modelo (Q) . Quanto ao tempo de execução, em 15 instâncias o modelo (Q) foi melhor que os algoritmos

GC_1^* e GC_2^* . Em apenas uma instância, minla-n10-t0.400-s1, ambos GC_1^* e GC_2^* , foram melhores que o modelo (Q). Por fim, em 4 instâncias prevaleceu o empate.

Fazendo um comparativo com os outros algoritmos de geração de colunas, podemos notar que em 13 instâncias o número de iterações de GC_1^* é menor que GC_1 , e em 6 instâncias o número de iterações de GC_1 foi menor. Quanto ao número de subproblemas resolvidos pelo B&B do CPLEX, em 15 instâncias o número de nós resolvidos por GC_1^* foi menor que o de GC_1 . Em 4 instâncias o número de nós resolvidos por GC_1 foi menor. Para o tempo de execução, tivemos que em 13 instâncias o algoritmo GC_1^* foi melhor, e em 5 instâncias o GC_1 foi quem obteve melhor resultado. Em apenas uma instância, minla-n10-t0.600-s2, ocorreu empate. O algoritmo GC_2^* obteve tempo de execução menor que GC_2 em 13 instâncias. Em 6 instâncias o algoritmo GC_2 conseguiu melhor desempenho que GC_2^* . Com relação ao número de nós, em 14 instâncias o número de nós resolvidos por GC_2^* foi menor que GC_2 . Em 5 instâncias o número de nós resolvidos por GC_2 foi menor. Por fim, em 13 instâncias o número de iterações de GC_2^* foi menor que GC_2 . Em 6 instâncias o número de iterações de GC_2 foi menor que GC_2^* .

Tanto o algoritmo GC_1^* quanto GC_2^* excederam o tempo limite de 5 horas para a instância GraphNug-n-23-t5 da classe I_B e para todas as instâncias da Tabela 12. Com a introdução de (2.74) os algoritmos GC_1^* e GC_2^* resolveram todas as outras instâncias da classe I_B . Além disso, se mostraram melhores que GC_1 e GC_2 nos três critérios avaliados. Entretanto, o modelo (Q) ainda foi mais eficiente que os dois algoritmos.

Tabela 16 – Resultados dos algoritmos de GC_1^* e GC_2^* - instâncias da classe I_A .

Instância	GC_1^*			GC_2^*			(Q)		
	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)
minla-n10-t0.200-s1	1.032	36.082	1	1.032	36.082	1	233	15.049	1
minla-n10-t0.200-s2	1.051	38.435	1	1.051	38.435	1	94	8.371	0
minla-n10-t0.300-s1	4.948	194.743	2	4.948	194.743	2	701	28.663	0
minla-n10-t0.300-s2	1.832	74.605	1	1.832	74.605	1	120	9.543	0
minla-n10-t0.400-s1	2.301	85.625	1	2.301	85.625	1	4.050	123.793	2
minla-n10-t0.400-s2	1.321	54.789	0	1.321	54.789	0	467	24.577	0
minla-n10-t0.500-s1	2.739	125.742	1	2.739	125.742	1	4.332	168.918	1
minla-n10-t0.500-s2	3.453	164.567	1	3.453	164.567	1	4.661	126.023	1
minla-n10-t0.600-s1	6.443	350.071	6	6.665	383.563	5	1.479	58.784	1
minla-n10-t0.600-s2	1.048	58.473	2	1.500	83.727	2	2.012	80.289	1
minla-n10-t0.700-s1	4.568	233.345	7	3.362	152.706	4	1.838	87.498	1
minla-n10-t0.700-s2	6.281	145.776	18	7.016	230.006	11	2.708	94.695	1
minla-n10-t0.800-s1	1.532	73.587	5	5.810	299.317	7	1.268	76.638	1
minla-n10-t0.800-s2	2.148	105.806	4	2.576	152.844	4	876	50.854	0

Fonte: Produzido pelos autores.

4.5.3 Modelo quadrático (P')

A partir das Tabelas 18 e 19 podemos notar que, com relação ao tempo de execução, em 4 instâncias o modelo (P') foi melhor que o modelo (Q). Em 7 instâncias o modelo (Q) obteve tempo de execução menor que o do modelo (P'). Nas outras 9 instâncias os

Tabela 17 – Resultados dos algoritmos de GC_1^* e GC_2^* - instâncias da classe I_B .

Instância	GC_1^*			GC_2^*			(Q)		
	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)
GraphNug-n-12-t5	1.855	93.433	2	1.855	93.433	3	28	4.620	0
GraphNug-n-15-t5	7.663	643.099	17	7.663	643.099	17	836	170.749	3
GraphNug-n-16-t6	9.839	2.321.906	174	9.839	2.321.906	169	639	201.245	3
GraphNug-n-17-t6	12.277	3.885.399	430	12.277	3.885.399	277	574	190.723	5
GraphNug-n-20-t5	14.435	4.326.139	570	14.435	4.326.139	391	10.143	3.343.838	139
GraphNug-n-23-t5	*	*	*	*	*	*	321.554	95.474.437	5.981

* Execução abortada depois que o limite de 5 horas foi alcançado.

Fonte: Produzido pelos autores.

tempos de execução de ambos os modelos (P') e (Q) foram iguais. Quanto ao número de subproblemas resolvidos pelo B&B do CPLEX, em 14 instâncias o número de nós resolvidos pelo modelo (P') foi menor que o do modelo (Q). Em 6 instâncias o número de nós resolvidos pelo modelo (Q) foi menor que o do modelo (P'). Já em termos de número de iterações, em 16 instâncias o número de iterações do modelo (P') foi menor que o do modelo (Q). Em 4 instâncias o número de iterações do modelo (Q) foi menor que o do modelo (P'). Comparando os resultados com o modelo (P), temos que em 6 instâncias o modelo (P') obteve tempo de execução menor que o modelo (P). Em 9 instâncias o modelo (P) foi quem obteve melhor desempenho. Em 5 instâncias prevaleceu o empate. Com relação ao número de iterações, em 10 instâncias o número de iterações do modelo (P') foi menor que o do modelo (P). Em 10 instâncias o número de iterações do modelo (P) foi menor. Quanto ao número de subproblemas resolvidos, em 10 instâncias o modelo (P') resolveu menos subproblemas que (P). Em 9 instâncias o número de subproblemas resolvidos por (P) foi menor. Em uma instâncias, minla-n10-t0.200-s2, ocorreu empate.

Nos quesitos número de iterações e subproblemas resolvidos, o modelo (P') foi melhor que o modelo (Q). Já no que diz respeito ao tempo de execução, o modelo (Q) obteve melhores resultados que (P'). Ambos os modelos, (P') e (P) tiveram resultados semelhantes. O modelo (P') resolveu menos subproblemas que (P), porém precisou de mais tempo para resolver as instâncias, além disso, tiveram o mesmo desempenho para o número de iterações. Portanto, temos que a inclusão de (2.74) em (P) não gerou uma melhora significativa.

4.5.4 Branch and Bound

Refizemos os experimentos com os algoritmos B_1 e B_2 considerando as desigualdades (2.74) em (Q). Denominamos esses algoritmos como B_1^* e B_2^* , respectivamente. Como pode ser observado nas Tabelas 20 e 21, em 11 instâncias o número de subproblemas resolvidos pelo algoritmo B_1^* foi menor que o do modelo (Q). E em 4 instâncias o número de nós resolvidos por Q foi menor que o do algoritmo B_1^* . Em 9 instâncias o número nós resolvidos pelo algoritmo B_2^* foi menor do que o do modelo (Q). Em 6 instâncias o número de nós resolvidos por Q foi menor. Quanto ao tempo de execução, em todas as instâncias o tempo

Tabela 18 – Resultados do modelo (P') com inclusão de (2.74) em (P) - instâncias da classe I_A .

Instância	(P')			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
minla-n10-t0.200-s1	555	20.360	1	233	15.049	1
minla-n10-t0.200-s2	0	736	0	94	8.371	0
minla-n10-t0.300-s1	2.893	94.248	2	701	28.663	0
minla-n10-t0.300-s2	0	995	0	120	9.543	0
minla-n10-t0.400-s1	617	27.720	0	4.050	123.793	2
minla-n10-t0.400-s2	236	7.148	0	467	24.577	0
minla-n10-t0.500-s1	652	24.852	1	4.332	168.918	1
minla-n10-t0.500-s2	615	26.315	0	4.661	126.023	1
minla-n10-t0.600-s1	437	16.900	0	1.479	58.784	1
minla-n10-t0.600-s2	238	8.875	1	2.012	80.289	1
minla-n10-t0.700-s1	909	31.399	1	1.838	87.498	1
minla-n10-t0.700-s2	2.045	84.692	2	2.708	94.695	1
minla-n10-t0.800-s1	574	21.803	1	1.268	76.638	1
minla-n10-t0.800-s2	355	17.488	0	876	50.854	0

Fonte: Produzido pelos autores.

Tabela 19 – Resultados do modelo (P') com inclusão de (2.74) em (P) - instâncias da classe I_B .

Instância	(P')			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
GraphNug-n-12-t5	781	34.342	2	28	4.620	0
GraphNug-n-15-t5	1.406	106.371	6	836	170.749	3
GraphNug-n-16-t6	672	243.524	14	639	201.245	3
GraphNug-n-17-t6	74	51.752	9	574	190.723	5
GraphNug-n-20-t5	12.185	2.231.843	273	10.143	3.343.838	139
GraphNug-n-23-t5	6.560	2.684.208	934	321.554	95.474.437	5.981

Fonte: Produzido pelos autores.

de execução do modelo (Q) foi melhor que o dos algoritmos B_1^* e B_2^* . É importante ressaltar que em 5 instâncias, ambos os algoritmos B_1^* e B_2^* , ultrapassaram o limite de tempo de 5 horas e tiveram sua execução abortada.

Fazendo um comparativo com os outros algoritmos de B&B, podemos notar que em todas as instâncias o algoritmo B_1^* obteve melhores resultados que B_1 , tanto no número de subproblemas resolvidos quanto no tempo de execução. O algoritmo B_2^* resolveu menos subproblemas que B_2 em 13 instâncias. Em 2 instâncias o número de nós resolvidos por B_2 foi menor que o do algoritmo B_2^* . Quanto ao tempo de execução, em 13 instâncias o algoritmo B_2^* foi mais eficiente que B_2 . Em 2 instâncias o algoritmo B_2 foi melhor.

Tanto o algoritmo B_1^* quanto B_2^* excederam o tempo limite de 5 horas para 5 instâncias da classe I_B e para todas as instâncias da Tabela 12. Com a introdução de (2.74) os algoritmos B_1^* e B_2^* se mostraram melhores que B_1 e B_2 , respectivamente. No entanto, o modelo (Q) ainda foi mais eficiente que os dois algoritmos.

Tabela 20 – Resultados dos algoritmos de B_1^* e B_2^* com a inclusão de (2.74) - instâncias da classe I_A .

Instância	B_1^*		B_2^*		(Q)	
	bb	$t(s)$	bb	$t(s)$	bb	$t(s)$
minla-n10-t0.200-s1	470	4	978	8	233	1
minla-n10-t0.200-s2	426	5	1.270	12	94	0
minla-n10-t0.300-s1	1.364	14	4.202	39	701	0
minla-n10-t0.300-s2	196	2	1.394	11	120	0
minla-n10-t0.400-s1	376	3	662	5	4.050	2
minla-n10-t0.400-s2	278	2	892	7	467	0
minla-n10-t0.500-s1	476	3	996	7	4.332	1
minla-n10-t0.500-s2	488	4	1.106	8	4.661	1
minla-n10-t0.600-s1	306	2	844	5	1.479	1
minla-n10-t0.600-s2	238	2	476	3	2.012	1
minla-n10-t0.700-s1	800	5	838	5	1.838	1
minla-n10-t0.700-s2	2.442	14	1.672	10	2.708	1
minla-n10-t0.800-s1	842	5	407	2	1.268	1
minla-n10-t0.800-s2	908	6	475	2	876	0

Fonte: Produzido pelos autores.

Tabela 21 – Resultados dos algoritmos de B_1^* e B_2^* com a inclusão de 2.74 - instâncias da classe I_B .

Instância	B_1^*		B_2^*		(Q)	
	bb	$t(s)$	bb	$t(s)$	bb	$t(s)$
GraphNug-n-12-t5	2.470	45	4.854	109	4.620	0
GraphNug-n-15-t5	*	*	*	*	170.749	3
GraphNug-n-16-t6	*	*	*	*	201.245	3
GraphNug-n-17-t6	*	*	*	*	190.723	5
GraphNug-n-20-t5	*	*	*	*	3.343.838	139
GraphNug-n-23-t5	*	*	*	*	95.474.437	5.981

* Execução abortada depois que o limite de 5 horas foi alcançado.

Fonte: Produzido pelos autores.

4.6 Analisando os resultados

Nesta análise, consideramos somente os modelos (A) , (M') , (Q) , (P) e (Q') . Além disso, usamos o esquema de votação proposto por (BORDA, 1781). Nesse método, cada modelo recebe uma pontuação de acordo com seu desempenho em cada um dos critérios avaliados (bb , $iter$, $t(s)$). Para cada um desses critérios, o modelo com melhor desempenho receberá 5 pontos, o segundo melhor 4 pontos, e assim sucessivamente. Se dois os mais modelos tiverem o mesmo desempenho em algum dos critérios analisados, então eles ficarão com a mesma pontuação. As Tabelas 22 e 23 contêm as pontuações dos modelos para as instâncias da classe I_A e I_B , respectivamente.

A Tabela 24 apresenta a soma total das pontuações dos modelos para cada um dos critérios analisados. É importante lembrar que quanto maior a pontuação, melhor foi o

Tabela 22 – Pontuação - instâncias da classe I_A .

Instância	(A)			(M')			(Q)			(P)			(Q')		
	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)
minla-n10-t0.200-s1	5	3	3	3	1	2	4	2	4	5	4	5	5	5	4
minla-n10-t0.200-s2	5	2	2	3	1	3	4	3	5	5	4	4	5	5	4
minla-n10-t0.300-s1	5	4	3	1	1	2	2	2	5	4	3	4	3	5	4
minla-n10-t0.300-s2	5	4	3	2	1	2	3	2	5	4	3	5	5	5	5
minla-n10-t0.400-s1	5	3	2	1	1	3	2	2	4	3	4	5	4	5	5
minla-n10-t0.400-s2	5	4	2	2	1	3	4	3	5	3	2	4	5	5	4
minla-n10-t0.500-s1	5	3	2	1	1	3	2	2	5	3	4	5	4	5	5
minla-n10-t0.500-s2	5	3	2	1	1	3	2	2	4	3	4	4	4	5	5
minla-n10-t0.600-s1	5	2	2	1	1	3	2	3	4	3	4	5	4	5	5
minla-n10-t0.600-s2	5	3	2	1	1	3	2	2	4	3	4	4	4	5	5
minla-n10-t0.700-s1	5	3	3	1	1	4	2	2	5	4	5	5	3	4	5
minla-n10-t0.700-s2	5	3	2	1	1	3	3	4	4	4	5	5	2	2	4
minla-n10-t0.800-s1	5	3	2	1	1	3	3	2	4	4	5	5	2	4	4
minla-n10-t0.800-s2	5	4	2	1	1	3	3	2	5	4	5	4	2	3	4

Fonte: Produzido pelos autores.

Tabela 23 – Pontuação - instâncias da classe I_B .

Instância	(A)			(M')			(Q)			(P)			(Q')		
	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)	bb	iter	t(s)
GraphNug-n-12-t5	5	5	3	3	2	4	4	1	5	5	3	5	5	4	5
GraphNug-n-15-t5	5	5	3	2	1	2	3	2	4	5	4	5	4	3	4
GraphNug-n-16-t6	4	1	1	3	4	4	2	2	3	5	5	5	1	3	2
GraphNug-n-17-t6	4	1	2	3	4	4	2	3	4	5	5	5	1	2	3
GraphNug-n-20-t5	5	2	1	1	1	2	2	3	3	4	4	5	3	5	4
GraphNug-n-23-t5	5	5	2	1	1	1	2	2	3	3	3	4	4	4	5

Fonte: Produzido pelos autores.

desempenho do modelo no aspecto investigado. No quesito número de iterações, o modelo (Q') foi quem obteve o melhor resultado. Já com relação ao número de nós avaliados na árvore de B&B do CPLEX, o modelo (A) obteve a maior pontuação. Por fim, o modelo (P) foi quem obteve o melhor desempenho relativo ao tempo de execução.

Tabela 24 – Pontuação total de cada um dos modelos para cada critério estudado.

	Modelo (A)	Modelo (M')	Modelo (Q)	Modelo (P)	Modelo (Q')
<i>Time</i>	44	57	85	93	86
<i>Nodes</i>	98	33	53	79	70
<i>Iter</i>	63	27	46	80	84

Fonte: Produzido pelos autores.

Realizamos também uma análise através da média aritmética. As Tabelas 25 e 26 apresentam os resultados para as instâncias da classe I_A e I_B , respectivamente. A instância GraphNug-n-23-t5 da classe I_B não foi considerada nessa análise, pois o modelo (M') não conseguiu resolvê-la no tempo limite de 5 horas. Os resultados dessa análise reforçam os resultados obtidos com o método de votação descrito anteriormente.

Tabela 25 – Análise via média para as instâncias da classe I_A .

	Modelo (Q)	Modelo (P)	Modelo (Q')	Modelo (A)	Modelo (M')
<i>Time(s)</i>	0,71	0,64	0,71	18,71	8,78
<i>Nodes</i>	1.774,21	1.095,07	800,35	10,14	8.293,92
<i>Iter</i>	68.121,07	32.060,35	24.999,21	61.630,00	396.482,57

Fonte: Produzido pelos autores.

Tabela 26 – Análise via média para as instâncias da classe I_B .

	Modelo (Q)	Modelo (P)	Modelo (Q')	Modelo (A')	Modelo (M')
<i>Time(s)</i>	30,00	11,00	27,00	17.275,20	1.170,40
<i>Nodes</i>	2.444,00	1.334,40	2.106,80	3,80	21.855,60
<i>Iter</i>	782.235,00	261.440,60	259.701,80	1.084.271,80	2.209.964,80

Fonte: Produzido pelos autores.

5 Conclusão e Trabalhos Futuros

Neste trabalho, implementamos e validamos o modelo quadrático proposto por (ANDRADE; BONATES, 2015) para o problema do arranjo linear mínimo. Esse modelo apresenta o menor número de variáveis e restrições que qualquer outro modelo presente na literatura do MinLA. Melhoramos o novo modelo misto com a introdução de novas desigualdades válidas para o problema. Realizamos alguns experimentos computacionais com o novo modelo e o comparamos com outros dois propostos por (AMARAL, 2009) e (MOEINI; GUEYE; LOYAL, 2014). Implementamos também um algoritmo de B&B e de geração de colunas para o problema.

Experimentos computacionais realizados em uma bateria de 27 instâncias de densidades variadas mostram que, de uma forma geral, o novo modelo (Q) proposto é mais eficiente que todos os demais encontrados na literatura. Obtivemos também bons resultados usando o modelo quadrático (P), no entanto, o mesmo não ocorreu para os algoritmos de B&B e de geração de colunas.

Observamos que a inclusão da desigualdade (2.74) em (Q) permitiu encontrar soluções ótimas para grafos densos com até 26 vértices. No modelo (P), a introdução dessa desigualdade não gerou melhorias significativas. Ambos os algoritmos, B&B e geração de colunas, obtiveram melhores resultados com a adição de (2.74). No entanto, o modelo (Q) ainda foi mais eficiente que os dois.

Como trabalhos futuros, queremos explorar a estrutura geométrica do poliedro do problema para fortalecer os modelos propostos. Continuaremos o estudo de desigualdades válidas na tentativa de melhorar a relaxação linear do problema. Pretendemos aplicar o novo modelo no problema de escalonamento de tarefas em uma máquina simples (ADOLPHSON, 1977; RAVI; AGRAWAL; KLEIN, 1991). E por fim, planejamos incorporar técnicas de programação por restrições na resolução do problema e investigar se o modelo descrito em (GOEMANS, 2009) pode ser aproveitado para fortalecer o nosso.

Referências

- ADAMS, W. P.; SHERALI, H. D. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, INFORMS, v. 32, n. 10, p. 1274–1290, 1986. Citado na página 41.
- ADOLPHSON, D. L. Single machine job sequencing with precedence constraints. *SIAM Journal on Computing*, SIAM, v. 6, n. 1, p. 40–54, 1977. Citado 2 vezes nas páginas 25 e 73.
- AMARAL, A. R. A mixed 0-1 linear programming formulation for the exact solution of the minimum linear arrangement problem. *Optimization Letters*, Springer, v. 3, n. 4, p. 513–520, 2009. Citado 10 vezes nas páginas 23, 27, 35, 40, 41, 42, 47, 57, 58 e 73.
- ANDRADE, R.; BONATES, T. O. New insights on minimum linear arrangements. *Universidade federal do Ceará, Departamento de Estatística e Matemática*, 2015. Citado 9 vezes nas páginas 23, 35, 43, 44, 45, 46, 47, 49 e 73.
- BORDA, J. C. de. Mémoire sur les élections au scrutin. *Histoire de L'Academie Royale des Sciences*, 1781. Citado na página 69.
- BURKARD, R. E.; KARISCH, S. E.; RENDL, F. Qaplib—a quadratic assignment problem library. *Journal of Global optimization*, Springer, v. 10, n. 4, p. 391–403, 1997. Citado na página 57.
- CAPRARA, A.; LETCHFORD, A. N.; GONZÁLEZ, J.-J. Decorous lower bounds for minimum linear arrangement. *INFORMS Journal on Computing*, INFORMS, v. 23, n. 1, p. 26–40, 2011. Citado na página 26.
- CHEN, P. P.-S. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, ACM, v. 1, n. 1, p. 9–36, 1976. Citado na página 26.
- CHUNG, F. R. K. *Selected topics in graph theory*. [S.l.]: Academic Press, 1988. ISBN 978-0-120-86203-0. Citado na página 27.
- DÍAZ, J.; PETIT, J.; SERNA, M. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, ACM, v. 34, n. 3, p. 313–356, 2002. Citado na página 23.
- GANE, C. P.; SARSON, T. *Structured systems analysis: Tools and techniques*. *McDonnell Douglas Systems Integration Company*, 1977. Citado na página 26.
- GAREY, M. R.; JOHNSON, D. S.; STOCKMEYER, L. Some simplified NP-complete graph problems. *Theoretical computer science*, Elsevier, v. 1, n. 3, p. 237–267, 1976. Citado 2 vezes nas páginas 23 e 26.
- GOEMANS, M. X. Smallest compact formulation for the permutahedron. *Mathematical Programming*, Springer, p. 1–7, 2009. Citado na página 73.
- HANAN, M.; KURTZBERG, J. M. A review of the placement and quadratic assignment problems. *Siam Review*, SIAM, v. 14, n. 2, p. 324–342, 1972. Citado na página 25.

- HARPER, L. H. Optimal assignments of numbers to vertices. *Journal of the Society for Industrial and Applied Mathematics*, JSTOR, p. 131–135, 1964. Citado na página 25.
- KIRKPATRICK, S.; VECCHI, M. P.; GELATT, J. Optimization by simulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983. Citado na página 32.
- METROPOLIS, N. et al. Equation of state calculations by fast computing machines. *The journal of chemical physics*, AIP Publishing, v. 21, n. 6, p. 1087–1092, 1953. Citado na página 30.
- MOEINI, M.; GUEYE, S.; LOYAL, S. M. A new mathematical model for the minimum linear arrangement problem. *ICORES - Angels - France*, p. 57–62, 2014. Citado 10 vezes nas páginas 23, 35, 36, 37, 38, 39, 40, 47, 57 e 73.
- NUGENT, C. E.; VOLLMANN, T. E.; RUML, J. An experimental comparison of techniques for the assignment of facilities to locations. *Operations research*, INFORMS, v. 16, n. 1, p. 150–173, 1968. Citado na página 57.
- PETIT, J. *Layout Problems*. Tese (Doutorado) — Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, 2001. Citado 4 vezes nas páginas 27, 29, 30 e 32.
- PETIT, J. Experiments on the minimum linear arrangement problem. *J. Exp. Algorithmics*, ACM, New York, NY, USA, v. 8, dez. 2003. ISSN 1084-6654. Disponível em: <<http://doi.acm.org/10.1145/996546.996554>>. Citado 3 vezes nas páginas 23, 25 e 27.
- RAVI, R.; AGRAWAL, A.; KLEIN, P. Ordering problems approximated: single-processor scheduling and interval graph completion. *Lecture Notes in Computer Science*, p. 751–762, 1991. Citado 2 vezes nas páginas 25 e 73.
- SCHWARZ, R. *A branch-and-cut algorithm with betweenness variables for the linear arrangement problems*. Tese (Doutorado) — Heidelberg University, Faculty of Mathematics and Informatics, Heidelberg, 2010. Citado 4 vezes nas páginas 23, 26, 27 e 35.
- SEITZ, H. *Contributions to the minimum linear arrangement problem*. Tese (Doutorado) — Heidelberg University, Faculty of Mathematics and Informatics, Heidelberg, 2010. Citado 5 vezes nas páginas 23, 26, 27, 35 e 43.
- VANNELLI, A.; ROWAN, G. An eigenvector-based approach for multistack vlsi layout. In: *Proceedings of the midwest symposium on circuits and systems*. [S.l.: s.n.], 1986. v. 29, p. 135–139. Citado na página 26.
- WOLSEY, L. A.; NEMHAUSER, G. L. Integer and combinatorial optimization. *Wiley1 ed ed*, 1999. Citado na página 36.

Apêndices

APÊNDICE A – Resultados extras

A.1 Resultados do modelo (Q) usando *Mip Start*

A partir dos resultados apresentados nas Tabelas 27 e 28, podemos notar que em 11 instâncias o número de iterações do modelo (Q) com *Mip Start* foi menor que o modelo (Q) puro. Em 9 instâncias o número de iterações do modelo (Q) puro foi menor. Quanto ao número de subproblemas resolvidos pelo B&B do CPLEX, em 10 instâncias o número de nós resolvidos pelo modelo (Q) com *Mip Start* foi menor que o modelo (Q) puro. Nas outras 10 instâncias o número de nós resolvidos pelo modelo (Q) puro foi menor. Por fim, quanto ao tempo de execução, em 7 instâncias o modelo (Q) com *Mip Start* foi melhor. Em apenas uma instância, GraphNug-n-20-t5, o modelo (Q) puro obteve tempo de execução menor. Nas outras 12 instâncias prevaleceu o empate. Em nossos experimentos utilizando *Mip Start* no modelo (Q) , observamos que os ganhos obtidos em relação ao modelo Q puro não foram tão significativos.

Tabela 27 – Resultados do modelo (Q) com *Mip Start* - instâncias da classe I_A .

Instância	(Q) com <i>Mip Start</i>			(Q)		
	bb	iter	t(s)	bb	iter	t(s)
minla-n10-t0.200-s1	300	15.441	0	233	15.049	1
minla-n10-t0.200-s2	91	7.846	0	94	8.371	0
minla-n10-t0.300-s1	573	29.213	0	701	28.663	0
minla-n10-t0.300-s2	123	9.399	0	120	9.543	0
minla-n10-t0.400-s1	4.180	169.556	1	4.050	123.793	2
minla-n10-t0.400-s2	520	30.998	0	467	24.577	0
minla-n10-t0.500-s1	4.437	131.242	1	4.332	168.918	1
minla-n10-t0.500-s2	3.524	108.564	1	4.661	126.023	1
minla-n10-t0.600-s1	1.945	76.980	1	1.479	58.784	1
minla-n10-t0.600-s2	2.361	99.367	1	2.012	80.289	1
minla-n10-t0.700-s1	1.971	106.110	0	1.838	87.498	1
minla-n10-t0.700-s2	3.078	134.920	1	2.708	94.695	1
minla-n10-t0.800-s1	966	50.758	0	1.268	76.638	1
minla-n10-t0.800-s2	1.232	57.633	0	876	50.854	0

Fonte: Produzido pelos autores.

A.2 Resultados do modelo (P) usando *Mip Start*

Realizamos alguns testes com o modelo quadrático (P) . Analisando as instâncias da classe I_A e I_B (Capítulo 4), observamos que em relação ao tempo de execução, em 5 instâncias o modelo (P) puro foi melhor que o modelo (P) com *Mip Start*. Em 2 instâncias o modelo (P) com *Mip Start* obteve tempo de execução menor que o do modelo (P) puro.

Tabela 28 – Resultados do modelo (Q) com *Mip Start* - instâncias da classe I_B .

Instância	(Q) com <i>Mip Start</i>			(Q)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
GraphNug-n-12-t5	0	369	0	28	4.620	0
GraphNug-n-15-t5	412	88.914	2	836	170.749	3
GraphNug-n-16-t6	636	197.480	3	639	201.245	3
GraphNug-n-17-t6	395	168.945	4	574	190.723	5
GraphNug-n-20-t5	9.651	3.265.311	140	10.143	3.343.838	139
GraphNug-n-23-t5	246.720	72.061.503	5.087	321.554	95.474.437	5.981

Fonte: Produzido pelos autores.

Nas outras 13 instâncias os tempos de execução de ambos os modelos (P) e (P) com *Mip Start* foram iguais. Quanto ao número de subproblemas resolvidos pelo B&B do CPLEX, em 12 instâncias o número de nós resolvidos pelo modelo (P) puro foi menor que o do modelo (P) com *Mip Start*. Em 4 instâncias o número de nós resolvidos pelo modelo (P) com *Mip Start* foi menor que o do modelo (P) puro. Em 4 instâncias prevaleceu o empate. Já em termos de número de iterações, em 12 instâncias o número de iterações do modelo (P) puro foi menor que o do modelo (P) com *Mip Start*. Em 7 instâncias o número de iterações do modelo (P) com *Mip Start* foi menor. Em apenas uma instância, GraphNug-n-12-t5, o número de iterações de ambos os modelos foram iguais.

Tabela 29 – Resultados do modelo (P) com *Mip Start* - instâncias da classe I_B .

Instância	(P) com <i>Mip Start</i>			(P)		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
GraphNug-n-12-t5	0	1.115	0	0	1.115	0
GraphNug-n-15-t5	0	1.919	1	0	2.308	1
GraphNug-n-16-t6	20	8.408	1	0	3.264	1
GraphNug-n-17-t6	0	3.478	2	0	3.195	1
GraphNug-n-20-t5	6.355	2.040.269	60	6.672	1.297.321	52
GraphNug-n-23-t5	314.038	82.720.310	5.782	181.153	62.023.724	5.057

Fonte: Produzido pelos autores.

Portanto, a partir dos nossos experimentos utilizando *Mip Start* no modelo (P) , podemos afirmar que o modelo (P) se mostrou mais eficiente que o modelo (P) com *Mip Start*.

Tabela 30 – Resultados do modelo (P) com *Mip Start* - instâncias da classe I_A .

Instância	<i>(P) com Mip Start</i>			<i>(P)</i>		
	<i>bb</i>	<i>iter</i>	<i>t(s)</i>	<i>bb</i>	<i>iter</i>	<i>t(s)</i>
minla-n10-t0.200-s1	249	12.312	0	0	632	0
minla-n10-t0.200-s2	0	910	1	0	912	1
minla-n10-t0.300-s1	661	28.922	0	596	27.519	1
minla-n10-t0.300-s2	0	741	0	106	8.719	0
minla-n10-t0.400-s1	4.185	133.542	1	3.327	106.295	1
minla-n10-t0.400-s2	834	36.086	1	945	40.940	1
minla-n10-t0.500-s1	2.808	50.364	0	2.762	61.675	1
minla-n10-t0.500-s2	2.942	72.917	1	1.701	42.607	1
minla-n10-t0.600-s1	1.088	31.914	1	926	22.580	0
minla-n10-t0.600-s2	1.520	37.721	1	1.564	40.063	1
minla-n10-t0.700-s1	1.267	35.561	1	790	20.183	1
minla-n10-t0.700-s2	2.293	57.886	1	1.795	40.385	0
minla-n10-t0.800-s1	500	15.746	0	408	13.900	0
minla-n10-t0.800-s2	0	371	1	411	22.435	1

Fonte: Produzido pelos autores.

APÊNDICE B – Um novo modelo trivial

Considere as seguintes variáveis binárias para todo $i, k \in \{1, \dots, n\}$:

$$x_{ik} = \begin{cases} 1, & \text{se o rótulo } k \text{ é atribuído ao vértice } i, \text{ isto é, } \pi_i = k; \\ 0, & \text{caso contrário.} \end{cases} \quad (\text{B.1})$$

Seja w_{ij} o peso da aresta $\{ij\} \in E$. O modelo é dado a seguir:

$$(T) \quad \min \quad \sum_{\{ij\} \in E} w_{ij} \quad (\text{B.2})$$

$$\text{s.a:} \quad \sum_{i=1}^n x_{ik} = 1 \quad \forall k \in \{1, \dots, n\} \quad (\text{B.3})$$

$$\sum_{k=1}^n x_{ik} = 1 \quad \forall i \in \{1, \dots, n\} \quad (\text{B.4})$$

$$w_{ij} \geq \sum_{k=1}^n kx_{ik} - \sum_{l=1}^n lx_{jl} \quad \forall \{ij\} \in E \quad (\text{B.5})$$

$$w_{ij} \geq \sum_{l=1}^n lx_{jl} - \sum_{k=1}^n kx_{ik} \quad \forall \{ij\} \in E \quad (\text{B.6})$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in \{1, \dots, n\} \quad (\text{B.7})$$

A restrição (B.3) garante que todo rótulo será atribuído a exatamente um vértice, e a restrição (B.4) garante que todo vértice terá exatamente um rótulo atribuído. Observe que $\sum_{l=1}^n lx_{jl}$ tem como resultado o rótulo l quando atribuído à um vértice j . Então, as restrições (B.5) e (B.6) garantem que $w_{ij} = \left| \sum_{l=1}^n lx_{jl} - \sum_{k=1}^n kx_{ik} \right|$, $\forall \{ij\} \in E$.

B.1 Resultados do modelo (T)

Os resultados são reportados nas Tabelas 31 e 32 a seguir. Nelas, podemos observar que em todas as instâncias o número de subproblemas resolvidos pelo B&B do CPLEX no modelo (Q) foi menor que o do modelo (T). Quanto ao número de iterações, em 9 instâncias o número de iterações do modelo (T) foi menor que o do modelo (Q). Também em 9 instâncias, o número de iterações do modelo (Q) foi menor que o do modelo (T). Com relação ao tempo de execução, em 8 instâncias o modelo (Q) foi melhor. Em 4 instâncias o modelo (T) obteve melhor resultado e em 6 instâncias prevaleceu o empate. Em 2 instâncias o modelo (T) ultrapassou o limite de tempo de 5 horas e teve sua execução interrompida. Não investigamos a fundo o modelo (T), porém nessa análise preliminar o modelo (Q) se mostrou mais eficiente.

Tabela 31 – Resultados do modelo (T) - instâncias da classe I_A .

Instância	(T)			(Q)		
	bb	iter	t(s)	bb	iter	t(s)
minla-n10-t0.200-s1	12.834	142.595	1	233	15.049	1
minla-n10-t0.200-s2	18.984	213.623	1	94	8.371	0
minla-n10-t0.300-s1	15.847	148.208	1	701	28.663	0
minla-n10-t0.300-s2	15.358	179.025	1	120	9.543	0
minla-n10-t0.400-s1	9.088	91.222	1	4.050	123.793	2
minla-n10-t0.400-s2	14.432	147.218	1	467	24.577	0
minla-n10-t0.500-s1	11.049	105.587	1	4.332	168.918	1
minla-n10-t0.500-s2	5.820	58.152	1	4.661	126.023	1
minla-n10-t0.600-s1	6.572	46.745	0	1.479	58.784	1
minla-n10-t0.600-s2	4.677	45.171	1	2.012	80.289	1
minla-n10-t0.700-s1	4.335	27.996	1	1.838	87.498	1
minla-n10-t0.700-s2	4.408	31.460	0	2.708	94.695	1
minla-n10-t0.800-s1	2.158	12.047	0	1.268	76.638	1
minla-n10-t0.800-s2	2.177	13.379	0	876	50.854	0

Fonte: Produzido pelos autores.

Tabela 32 – Resultados do modelo (T) - instâncias da classe I_B .

Instância	(T)			(Q)		
	bb	iter	t(s)	bb	iter	t(s)
GraphNug-n-12-t5	49.744	840.291	5	28	4.620	0
GraphNug-n-15-t5	15.216.478	249.325.948	3.706	836	170.749	3
GraphNug-n-16-t6	1.491.779	25.657.569	195	639	201.245	3
GraphNug-n-17-t6	32.766.798	622.339.974	11.798	574	190.723	5
GraphNug-n-20-t5	*	*	*	10.143	3.343.838	139
GraphNug-n-23-t5	*	*	*	321.554	95.474.437	5.981

* Execução abortada depois que o limite de 5 horas foi alcançado.

Fonte: Produzido pelos autores.