Recoloração Convexa de Caminhos

Karla Roberta P. Sampaio Lima

Tese apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Doutor em Ciências

Área de Concentração: Ciência da Computação

Orientadora: Profa. Dra. Yoshiko Wakabayashi

Durante a elaboração deste trabalho a autora recebeu auxílio financeiro do CNPq

São Paulo, dezembro de 2011

Recoloração Convexa de Caminhos

Esta tese contém as correções e alterações
sugeridas pela Comissão Julgadora durante a defesa
realizada por Karla Roberta Pereira Sampaio Lima em 16/11/2011.
O original encontra-se disponível no Instituto de
Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Profa. Dra. Yoshiko Wakabayashi (orientadora) - IME-USP
- Prof. Dr. Carlos Eduardo Ferreira - IME-USP
- Prof. Dr. Flávio Keidi Miyazawa - UNICAMP
- Prof. Dr. Manoel Bezerra Campêlo Neto - UFC
- Prof. Dra. Gordana Manić - UFABC

Agradecimentos

Deus, com toda sua benevolência e graça, sempre foi muito fiel em minha vida e é a Ele que quero agradecer e honrar em primeiro lugar. Com esse objetivo, gostaria de compartilhar e testemunhar em poucas palavras o que foi essa trajetória para mim.

Posso dizer que, mediante a fé, atravessei uma das mais difíceis etapas da minha vida, pois aprendi que "*a fé é a certeza das coisas que se esperam, e a convicção de fatos que não se vêem*". Mesmo em meio a tantas dificuldades e tribulações, ouvir a voz de Deus, por meio da sua palavra, me dizendo "*não temas porque eu sou o teu Deus, eu te fortaleço, eu te ajudo, e te sustento com minha destra fiel*", trouxe-me vida a cada manhã para seguir em frente. E ao longo desses anos, longe da família, eu posso dizer que perseverança foi a minha melhor decisão.

Em segundo lugar, agradeço à minha orientadora profa. Yoshiko, que sempre foi muito presente. Destacada pelos alunos por ser uma excelente professora, posso afirmar que ela também é uma excelente orientadora e amiga, me acrescentando muito nesse tempo. Não tenho como não expressar minha verdadeira admiração por ela e agradecer por tudo que ela fez por mim nesses anos. Não menos importantes, agradeço aos professores Carlinhos, Flávio Keidi, Manoel Campêlo, Gordana, Cris, Coelho, Feofiloff e Arnaldo, pois também aprendi muito com eles.

Agradeço ao meu querido esposo, por todo amor e apoio nesses últimos três anos juntos, ao meu pai, por um dia ter me ensinado a pescar (ser perseverante e paciente), à minha mãe, por toda sabedoria e pelas nossas intermináveis conversas ao telefone, à minha irmã, que é minha melhor amiga, às famílias Lima, Freire e Almeida pela torcida e presença na minha defesa, e a toda minha família e amigos de Goiânia, em especial,

ao meu amigo Walid.

Quantos aos amigos do IME, agradeço ao Luna pela amizade e força nos primeiros anos do doutorado, a todos os amigos do NUMEC, em especial ao Alexandre Freire pelas dicas. Para finalizar, agradeço por todo carinho e torcida, minhas grandes amigas: Giselle, Mariana, Tatiane, Juliana, Ana, Nathália, Vanessa, Marisa, Fanny, Priscila, Viviane, Fernanda e Susanna.

Resumo

O foco central desta tese é o desenvolvimento de algoritmos para o problema de recoloração convexa de caminhos. Neste problema, é dado um caminho cujos vértices estão coloridos arbitrariamente, e o objetivo é recolorir o menor número possível de vértices de modo a obter uma coloração convexa. Dizemos que uma coloração de um grafo é convexa se, para cada cor, o subgrafo induzido pelos vértices dessa cor é conexo. Sabe-se que este problema é NP-difícil.

Associamos a este problema um poliedro, e estudamos sua estrutura facial, com vistas ao desenvolvimento de um algoritmo. Mostramos várias inequações válidas para este poliedro, e provamos que várias delas definem facetas. Apresentamos um algoritmo de programação dinâmica que resolve em tempo polinomial o problema da separação para uma classe grande de inequações que definem facetas. Implementamos um algoritmo *branch-and-cut* baseado nesses resultados, e realizamos testes computacionais com instâncias geradas aleatoriamente. Apresentamos adicionalmente uma heurística baseada numa formulação linear que obtivemos.

Estudamos também um caso especial deste problema, no qual as instâncias consistem em caminhos coloridos, onde cada cor ocorre no máximo duas vezes. Apresentamos um algoritmo de $\frac{3}{2}$-aproximação para este caso, que é também NP-difícil. Para o caso geral, é conhecido na literatura um algoritmo de 2-aproximação.

Palavras-chave: recoloração convexa, caminho, algoritmo de aproximação, poliedro, faceta, branch-and-cut.

Abstract

The focus of this thesis is the design of algorithms for the convex recoloring problem on paths. In this problem, the instance consists of a path whose vertices are arbitrarily colored, and the objective is to recolor the least number of vertices so as to obtain a convex coloring. A coloring of a graph is convex if, for each color, the subgraph induced by the vertices of this color is connected. This problem is known to be NP-hard.

We associate a polyhedron to this problem and investigate its facial structure. We show various classes of valid inequalities for this polyhedron and prove that many of them define facets. We present a polynomial-time dynamic programming algorithm that solves, in polynomial time, the separation problem for a large class of facet-defining inequalities. We report on the computational experiments with a branch-and-cut algorithm that we propose for the problem. Additionally, we present a heuristic that is based on a linear formulation for the problem.

We also study a special case of this problem, restricted to instances consisting of colored paths in which each color occurs at most twice. For this case, which is also NP-hard, we present a $\frac{3}{2}$-approximation algorithm. For the general case, it is known a 2-approximation algorithm.

Keywords: convex recoloring, path, approximation algorithm, polyhedron, facet, branch-and-cut.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste trabalho, o conceito central é o de coloração convexa de grafos. Dizemos que uma coloração dos vértices de um grafo é convexa se, para cada cor, o subgrafo induzido pelos vértices dessa cor é conexo.

Investigamos o problema de recoloração convexa de grafos, para o caso especial de caminhos. Neste problema, é dado um caminho colorido nos vértices, e o objetivo é recolorir um número mínimo de vértices, de modo a obter uma coloração convexa. Este problema foi inicialmente proposto para árvores, em 2005. Desde então, apareceram estudos tanto para o caso especial de caminhos, bem como para grafos arbitrários. Sabe-se que este problema é NP-difícil, mesmo para caminhos onde cada cor ocorre no máximo duas vezes.

Para simplificar, denotamos o problema de recoloração convexa de grafos pela sigla **RCG** e o de caminhos pela sigla **RCC**. Usamos a sigla 2**-RCC** para denotar o caso especial de RCC em que cada cor ocorre no máximo duas vezes.

1.1 Aplicações

O problema de recoloração convexa de árvores foi introduzido por Moran e Snir [16], motivado por aplicações no estudo de árvores filogenéticas. Tais árvores representam similaridade e grau de parentesco entre espécies ou grupos de espécies. Mais especificamente, nas árvores filogenéticas, as folhas representam espécies existentes e os vértices internos correspondem a seus ancestrais, espécies extintas.

Na construção de tais árvores, são considerados certos atributos biológicos compartilhados pelas espécies em estudo. Um tal atributo é chamado *caractere*, e este pode ter diversos *estados*. Por exemplo, no estudo de seres vivos, a *forma como as espécies se locomovem* é um caractere, e alguns dos seus estados são: *rastejamento, locomoção na água e no ar*.

As árvores filogenéticas são deduzidas a partir de dados conhecidos, de tal modo que, as espécies que apresentam similaridade entre si, para um dado caractere que está sendo avaliado, têm vértices correspondentes mais próximos na árvore.

Usualmente, dada uma árvore filogenética, ao introduzir um novo caractere, este atribui a algumas espécies uma cor, dependendo do estado desse caractere na espécie. Obtemos, dessa forma, uma árvore com uma *coloração total* (se todo vértice tem cor), ou uma árvore com uma *coloração parcial*, ou seja, uma coloração que atribui cores somente a algum subconjunto próprio de vértices.

Neste contexto filogenético, amplia-se o conceito de coloração convexa para árvores filogenéticas parcialmente coloridas, definindo-se que uma coloração parcial é convexa se esta pode ser estendida a uma coloração total convexa.

Dado um conjunto de espécies relacionadas, a reconstrução filogenética consiste em obter uma árvore que melhor descreva a evolução desse conjunto de espécies. Neste contexto, dada uma árvore colorida, uma recoloração convexa de custo mínimo (a ser definida mais tarde) define uma filogenia perfeita, segundo Moran e Snir [16].

Maiores detalhes sobre a motivação filogenética que deu origem ao problema de recoloração convexa podem ser encontrados em [16], [5] e [1] .

1.2 Objetivos e contribuições

Nossa motivação para estudar o problema de recoloração convexa surgiu da constatação de que não havia na literatura abordagens baseadas em combinatória poliédrica para nenhuma das variantes desse problema. Os resultados conhecidos são na linha de algoritmos exatos (de enumeração), algoritmos-FPT e algoritmos de aproximação. Decidimos assim investigar esse problema, no caso de caminhos, através de métodos poliédricos.

Dada uma instância do problema RCC, que consiste num caminho colorido, associamos um poliedro $\mathcal{P}$ definido como o fecho convexo dos vetores de incidência das (re)colorações convexas desse caminho. Mostramos que este poliedro é o fecho convexo das soluções de uma formulação linear 0/1 para o problema RCC, e investigamos sua estrutura facial.

Exibimos várias classes de inequações válidas para o poliedro $\mathcal{P}$, e mostramos que várias delas definem facetas. Implementamos um algoritmo de programação dinâmica, polinomial, que resolve o problema da separação para uma classe grande de inequações que definem facetas, chamadas inequações de convexidade-generalizada.

Inicialmente, partimos de uma formulação linear relaxada, com uma classe simples de inequações, que chamamos de inequações de convexidade. Após várias tentativas

infrutíferas de "enxergar inequações que cortassem" as soluções fracionárias obtidas, foi apenas através da técnica de "fazer combinações de inequações e arredondamentos" que conseguimos encontrar inequações que definem facetas. Estas foram novamente combinadas, e deram origem a novas inequações, todas elas obtidas através de arredondamentos.

Implementamos um algoritmo *branch-and-bound* baseado nos resultados poliedrais encontrados. Percebemos, através de experimentos computacionais, que essa classe grande de inequações que definem facetas, quando adicionadas à formulação relaxada — à medida que são encontradas pelo algoritmo de separação — em geral produzem ou uma solução inteira, ou uma solução fracionária cujo valor está bem próximo do valor ótimo. Apresentamos os resultados computacionais obtidos com esse algoritmo e outros.

Estudamos também o problema RCC com vistas ao desenvolvimento de um algoritmo de aproximação, que tivesse um desempenho melhor do que o algoritmo de 2-aproximação, obtido por Moran e Snir [16]

Um algoritmo de aproximação caracteriza-se, em termos gerais, pela sua eficiência e garantia de desempenho. Essa garantia de desempenho refere-se a quão próximo a solução devolvida por tal algoritmo de aproximação está de uma solução ótima. Essa medida é dada em termos da razão dos valores dessas soluções.

Mais recentemente, em 2009, um caso mais restrito deste problema, que se refere a caminhos coloridos em que cada cor ocorre no máximo duas vezes, foi provado ser NP-difícil por Kanj e Kratsch [14]. Para este problema, 2-RCC, obtivemos um algoritmo de $\frac{3}{2}$-aproximação [15].

1.3 Organização do trabalho

No *Capítulo 2* apresentamos algumas definições básicas e estabelecemos a notação que adotamos neste texto.

No *Capítulo 3* mencionamos os principais resultados conhecidos para todas as variantes do problema de recoloração convexa de grafos. Exibimos também a prova apresentada por Kanj e Kratsch [14] de que o problema 2-RCC é NP-difícil.

Exibimos no *Capítulo 4* um algoritmo de $\frac{3}{2}$-aproximação para o problema 2-RCC. Apresentamos também o algoritmo de 2-aproximação, obtido por Moran e Snir [17], que foi a fonte de inspiração para desenvolvermos o algoritmo de $\frac{3}{2}$-aproximação [15].

O *Capítulo 5* contém os resultados teóricos sobre o poliedro que associamos ao problema RCC. Exibimos várias inequações válidas para esse poliedro, e provamos quais delas definem facetas.

No *Capítulo 6* concentram-se os resultados computacionais deste trabalho. Descrevemos um algoritmo de separação polinomial para uma classe grande de inequações que definem facetas. Mencionamos alguns detalhes sobre o algoritmo *branch-and-cut* que implementamos para o RCC. Apresentamos algumas tabelas com os dados dos experimentos realizados, incluindo aqueles obtidos com o algoritmo de 2-aproximação de Moran e Snir, e uma heurística gulosa que desenvolvemos, baseado numa relaxação linear.

Algumas considerações finais são apresentadas no *Capítulo 7*.

Capítulo 2

Preliminares

Neste capítulo, apresentamos alguns conceitos básicos sobre grafos, álgebra linear e combinatória poliédrica, mais com a finalidade de estabelecer a notação.

2.1 Teoria dos grafos

Os conceitos a serem apresentados aqui são básicos e podem ser encontrados em qualquer livro introdutório sobre grafos. Seguimos mais de perto a terminologia usada no livro de Bondy e Murty [7].

Um **grafo** G é um par ordenado (V, A), onde V é um conjunto finito de elementos chamados **vértices** e A é um conjunto de elementos chamados **arestas**, sendo que cada aresta é um par não-ordenado de vértices distintos. Quando conveniente, para referir ao conjunto de vértices (resp. arestas) de um grafo usamos a notação $V(G)$ (resp. $A(G)$).

Observamos que, na maioria dos textos, o objeto que definimos como grafo é chamado de grafos simples.

Se $\alpha = \{u, v\}$ é uma aresta, dizemos que α **incide** em u e em v, ou que u e v são seus **extremos** ou ainda que u e v são **adjacentes**. Também denotamos $\{u, v\}$ por uv.

Um **caminho** num grafo G é uma sequência de vértices distintos $P = (v_1, v_2, \ldots, v_k)$, tal que $v_i v_{i+1} \in A(G)$ para $i = 1, \ldots, k-1$. Neste caso, dizemos que $V(P) = \{v_1, \ldots, v_k\}$ e $A(P) = \{v_i v_{i+1} : i = 1, \ldots, k-1\}$. Dizemos que v_1 é o **início** e v_k é o **término** de P, e que P é um caminho de v_1 a v_k. O **comprimento** de um caminho é o número de seus vértices menos 1. Se $P = (v_1, v_2, \ldots, v_k)$ é um caminho num grafo, $k \geq 3$, e v_k e v_1 são adjacentes, então $C = (v_1, v_2, \ldots, v_k, v_1)$ é um **circuito**.

Um grafo G é **conexo** se para quaisquer vértices distintos u e v em G existe um caminho de u a v. Uma **árvore** é um grafo conexo sem circuitos.

Se dois grafos $G = (V, A)$ e $H = (W, B)$ são tais que $V \subseteq W$ e $A \subseteq B$, então G é dito um **subgrafo** de H. Se $X \subseteq V$, então o subgrafo de G **induzido** por X, denotado por $G[X]$, é o grafo cujo conjunto de vértices é X e cujo conjunto de arestas consiste

nas arestas de G com ambos os extremos em X. Denotamos por $G - W$ o subgrafo obtido de G removendo-se W, definido como sendo o grafo induzido por $V \setminus W$, isto é $G[V \setminus W]$. Se G é um caminho, então um subgrafo de G que é um caminho é chamado de **subcaminho**.

Um conjunto de vértices I em um grafo é **independente** se os vértices de I são dois a dois não-adjacentes.

2.2 Coloração, recoloração e convexidade

Uma **coloração** de um grafo $G = (V, A)$ é uma função $C : V \to \mathcal{C}$, onde $\mathcal{C}$ é um conjunto de cores. Ressaltamos que a coloração aqui definida não é uma coloração própria, como entendido na teoria dos grafos; trata-se de uma simples atribuição de cores, sem nenhuma restrição. Um **grafo colorido** é um par (G, C) que consiste em um grafo G e uma coloração C de G. Quando G é uma árvore (resp. caminho), referimo-nos à **árvore colorida** (resp. **caminho colorido**). Convencionamos aqui que, se C é uma coloração, então $\mathcal{C}$ denota o conjunto de cores usadas por C.

Uma coloração C de G é **convexa** se, para cada cor $i \in \mathcal{C}$, o conjunto dos vértices com a cor i induz um subgrafo conexo de G.

Quando a coloração de um grafo colorido (G, C) não é convexa, é desejável saber quão distante esta coloração está de uma que seja convexa (e que possa ser obtida com o menor número de alterações de cor). Tal distância é definida como **distância de recoloração** de (G, C). O processo de recolorir os vértices de um grafo colorido (G, C) consiste em simplesmente alterar a cor de um ou mais vértices de G. Neste processo, algumas cores da coloração inicialmente dada podem ser eliminadas. A nova coloração obtida, digamos C', é chamada uma **recoloração** de (G, C). Se ela é convexa, dizemos que é uma **recoloração convexa** de (G, C).

Se C' é uma recoloração de (G, C), definimos o **custo** de C', denotado por $\text{custo}_C(C')$, ou simplesmente $\text{custo}(C')$, como $|\{v \in V(G) : C'(v) \neq C(v)\}|$.

Se C^* é uma recoloração convexa de C e $\text{custo}_C(C^*) \leq \text{custo}_C(C')$ para qualquer recoloração convexa C' de C, então dizemos que C^* é uma **recoloração convexa ótima** de (G, C), e denotamos $\text{custo}_C(C^*)$ por $\text{opt}(G, C)$.

O estudo de recoloração convexa de grafos considera dois modelos: sem pesos nos vértices e com pesos nos vértices. O modelo sem pesos nos vértices é o que consideramos até agora e será tratado neste trabalho.

Para o modelo com pesos nos vértices, é dada uma função peso com valores reais não-negativos sobre os vértices do grafo.

Mais formalmente, neste modelo, a instância é uma tripla (G, C, w), onde $w : V(G) \to \mathbb{R}_{\geq 0}$ é uma função peso que atribui para cada vértice v um valor real não-negativo $w(v)$,

definido como o **peso** de v. Para um conjunto de vértices X, o **peso de X**, denotado por $w(X)$, é definido como $w(X) := \sum_{v \in X} w(v)$. Neste caso, o **custo de uma recoloração convexa** C' de (G, C) é $\text{custo}_C(C') = w(X(C'))$, onde $X(C') = \{v \in V(G) : C(v) \neq C'(v)\}$. Analogamente, define-se **recoloração convexa ótima** como sendo uma recoloração convexa de custo mínimo.

Claramente, o modelo sem pesos pode ser visto como um caso especial do modelo com pesos, no qual cada vértice tem peso unitário.

2.3 Algoritmos de aproximação

Seja $\mathcal{A}$ um algoritmo para um problema de otimização P, cujo objetivo é encontrar uma solução para cada instância I, que seja de menor custo possível.

Dizemos que $\mathcal{A}$ é uma **α-aproximação** para P se, para qualquer instância I do problema, o algoritmo $\mathcal{A}$ produz em tempo polinomial (no tamanho da instância I) uma solução $\mathcal{A}(I)$ tal que $\text{custo}(\mathcal{A}(I)) \leq \alpha \, \text{opt}(I)$, onde α é uma constante ou uma função que depende de I, e $\text{opt}(I)$ denota o custo de uma solução ótima para a instância I.

Problemas nos quais algum parâmetro k é fixado, são chamados de **problemas parametrizados**. Dizemos que um problema P é (ou pertence à classe) FPT (*Fixed Parameter Tractable*) com respeito ao parâmetro k, se existe um algoritmo de complexidade $O(f(k)\, m^{O(1)})$ para P, onde f é uma função de k que independe de m, o tamanho da instância considerada. Para simplificar, chamamos um tal algoritmo de **algoritmo FPT**.

Para simplificar, na notação de complexidade, escrevemos $O^*(f(n))$ em vez de $O(f(n)\, p(n)))$, onde p é um polinômio em n. Assim, por exemplo, a notação $O^*(2^n)$ significa $O(2^n\, p(n))$.

O leitor interessado em algoritmos de aproximação pode consultar o livro de Ausiello et al. [2] e o texto de Carvalho et al. [8]; para a parte de complexidade parametrizada, sugerimos o livro de Downey e Fellows [10].

2.4 Conceitos básicos sobre poliedros e programação linear

Um problema de programação linear (PL) consiste em um problema de maximizar (ou minimizar) uma função linear, que deve satisfazer um conjunto de restrições lineares. Estas restrições podem ser igualdades e/ou desigualdades.

Um **poliedro** $\mathcal{P} \subseteq \mathbb{R}^n$ é o conjunto solução de um sistema finito de inequações lineares; isto é $\mathcal{P} = \{x \in \mathbb{R}^n : Ax \leq b\}$, onde (A, b) é uma matriz $(m \times n) + 1$. Neste caso, também denotamos $\mathcal{P}$ por $\mathcal{P}(A, b)$.

Um problema de **programação linear inteira**, ou programa linear inteiro (PLI), é um PL em que as variáveis podem assumir somente valores inteiros. Quando esses valores

inteiros são apenas 0 ou 1, então nos referimos a um PL/01. Enquanto PL's podem ser resolvidos em tempo polinomial, PLI's são em geral NP-difíceis.

Formalmente, um programa linear inteiro é um PL da forma

$$\begin{array}{rrcl} \text{maximize} & cx & & \\ \text{sujeito a} & Ax & \leq & b \\ & x & \in & \mathbb{Z}^n_+ \end{array}$$

Dado um PLI de maximização, digamos $\mathcal{P}$, toda solução viável de $\mathcal{P}$ é um limite inferior para o valor da solução ótima. No entanto, existem problemas em que encontrar boas soluções viáveis é tão difícil quanto resolver o próprio PLI. Por outro lado, existem métodos com os quais podemos encontrar limites superiores para este problema. Um desses métodos consiste em resolver o problema que se obtém retirando-se a condição de integralidade, chamado de uma **relaxação linear** de $\mathcal{P}$.

Um problema $z^R = max\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ é uma **relaxação** de $z = max\{c(x) : x \in X \subseteq \mathbb{Z}^n\}$ se $X \subseteq T$ e $f(x) \geq c(x)$ para todo $x \in X$.

Um vetor $x \in \mathbb{R}^n$ é uma **combinação linear** dos vetores $x_1, \ldots, x_t \in \mathbb{R}^n$ se, para algum $\alpha = (\alpha_1, \ldots, \alpha_t) \in \mathbb{R}^t$, $x = \sum_{i=1}^t \alpha_i x_i$. Uma tal combinação linear é chamada

combinação afim se $\alpha_1 + \alpha_2 + \ldots + \alpha_t = 1$;
combinação cônica se $\alpha_1, \ldots, \alpha_t \geq 0$; e
combinação convexa se for afim e cônica.

Para um conjunto não-vazio $S \subseteq \mathbb{R}^n$, definimos o **fecho afim** dos elementos de S, denotado por **afim**(S), como o conjunto de todos os vetores que são combinação afim de um número finito de elementos de S. Define-se analogamente o conceito de **fecho convexo** de S, denotado por **conv**(S): o conjunto de todos os vetores que são combinação convexa de um número finito de vetores de S.

Um conjunto $S \subseteq \mathbb{R}^n$ é **linearmente independente** se para qualquer subconjunto finito $\{x_1, \ldots, x_t\}$ de S, sempre que $\sum_{i=1}^t \alpha_i x_i = 0$, onde $\alpha_i \in \mathbb{R}$, temos que $\alpha_1 = \alpha_2 = \ldots = \alpha_t = 0$. Similarmente, $S \subseteq \mathbb{R}^n$ é **afim independente** se para qualquer subconjunto finito $\{x_1, \ldots, x_t\}$ de S, $\sum_{i=1}^t \alpha_i x_i = 0$, onde $\alpha_i \in \mathbb{R}$ e $\sum_{i=1}^t \alpha_i = 0$, implica que $\alpha_1 = \alpha_2 = \ldots = \alpha_t = 0$.

Para $S \subseteq \mathbb{R}^n$, o **posto** de S, denotado por **posto**(S), é a cardinalidade de um maior subconjunto de S que é linearmente independente. Similarmente, **posto afim** de S, denotado por **posto-afim**(S), é a cardinalidade de um maior conjunto afim-independente contido em S.

A **dimensão** de um poliedro $\mathcal{P}$, denotado por **dim**$(\mathcal{P})$, é o número máximo de vetores afim independentes em $\mathcal{P}$ menos um. Um poliedro $\mathcal{P} \subseteq \mathbb{R}^n$ tem **dimensão plena** se $\dim(\mathcal{P}) = n$.

O seguinte resultado será usado no Capítulo 5.

Proposição 2.4.1 *Seja* $\mathcal{P} \subseteq \mathbb{R}^n$ *um poliedro tal que* $\text{afim}(\mathcal{P}) = \{x \in \mathbb{R}^n : Ax = b\}$. *Então*

$$\dim(\mathcal{P}) = n - \text{posto}(A).$$

Se $\mathcal{P}$ é um poliedro, dizemos que uma inequação $a^T x \leq x_0$ é uma **inequação válida** para $\mathcal{P}$ se $a^T w \leq x_0$ para todo $w \in \mathcal{P}$.

Dizemos que F é uma **face** de um poliedro $\mathcal{P}$ se $F = \{x \in \mathcal{P} : a^T x = a_0\}$ para alguma inequação válida $a^T x \leq a_0$ para $\mathcal{P}$. Uma face F de $\mathcal{P}$ é uma **faceta** se $\dim(F) = \dim(\mathcal{P}) - 1$. Se F é uma **faceta** de $\mathcal{P}$ e $F = \{x \in \mathcal{P} : \gamma x = \gamma_0\}$, dizemos que a inequação válida $\gamma x \leq \gamma_0$ **define** (ou **induz**) a faceta F.

Se $\mathcal{P}$ tem dimensão plena, então $\mathcal{P}$ tem uma descrição única minimal, dada pelas inequações que definem facetas de $\mathcal{P}$. Tais inequações são únicas, a menos de múltiplos escalares; e são necessárias, no sentido de que se uma delas for removida obtém-se um outro poliedro. Para poliedros de dimensão plena, esse fato facilita a prova de que certas inequações definem facetas. Quando o poliedro não tem dimensão plena, como é o caso do poliedro que estudamos neste trabalho, as inequações que são válidas para o poliedro não são expressas de forma única, pois podem resultar (de combinações cônicas de inequações e) de combinações lineares de equações que definem o fecho afim do poliedro. Isso torna um pouco mais trabalhosa a prova de que certas inequações válidas definem facetas.

Resultados e outros conceitos sobre poliedros, programas lineares, dualidade, *branch-and-bound* e *branch-and-cut* podem ser encontrados no texto de Ferreira e Wakabayashi [11] e no livros de Schrijver [22] e Wolsey [23].

Capítulo 3

Resultados conhecidos sobre o problema RCG

Neste capítulo, apresentamos os resultados conhecidos para o problema de recoloração convexa de grafos, denotado por RCG.

Os primeiros resultados para o problema de recoloração convexa apareceram para árvores e foram obtidos por Moran e Snir [18].

Esses estudos surgiram motivados por sua relação com o problema de construção de árvores filogenéticas. Mais recentemente, surgiram na literatura resultados para grafos arbitrários. Em 2007, Chor, Fellows, Ragan, Razgon, Rosamond e Snir [9] introduziram uma versão generalizada para grafos arbitrários. Nesta versão, o conceito de convexidade que mencionamos pode ser visto como um caso especial de um conceito mais geral, no qual cada cor pode induzir um certo número máximo de componentes conexos. Neste capítulo definiremos mais formalmente este conceito e os resultados a respeito.

Na prática, estes estudos se aplicam a redes de interação de proteínas, um problema bastante estudado na Biologia. Em 2008, Kammer e Tholey [13] estudaram o problema de recoloração convexa de grafos como um caso especial de problema de roteadores.

Mencionaremos alguns resultados que encontramos na literatura, alguns dos quais requerem o conhecimento de certos conceitos, que não apresentaremos aqui. Sugerimos ao leitor interessado que consulte o livro de Downey e Fellows [10] sobre (in)tratabilidade parametrizada, classes de complexidade parametrizada (FPT, $W[1]$), largura arbórea de um grafo, kernelização, etc.

3.1 Recoloração convexa de grafos arbitrários

Apresentamos nesta seção alguns resultados conhecidos para o problema de recoloração convexa de grafos. No que segue, r é um natural positivo.

Lembramos aqui alguns conceitos que foram definidos no Capítulo 2, e que serão usados aqui. Um grafo colorido é um par (G, C) que consiste em um grafo G e uma

coloração C de G. Convencionamos que se C é uma coloração, então $\mathcal{C}$ denota o conjunto de cores usadas por C.

Se C é uma coloração parcial de G, isto é C atribui cor a um subconjunto próprio de vértices de G, dizemos que (G, C) é um grafo parcialmente colorido. Os vértices que não recebem cor são chamados de vértices sem cor. Uma coloração parcial é dita ser convexa e esta pode ser estendida à uma coloração convexa total.

Vimos também que uma coloração C de G é convexa se o conjunto dos vértices de cada cor $i \in \mathcal{C}$ induz um subgrafo conexo de G.

> Problema r-Recoloração Convexa de Grafos Coloridos Parcialmente (r-RCGP)
> Instância: Um grafo G, um conjunto de cores $\mathcal{C}$, uma coloração parcial $C : V \to \mathcal{C}$, onde existem k vértices sem cor.
> Questão: É possível completar C à uma coloração total C' tal que cada cor induz um subgrafo com no máximo r componentes?

Observe, neste problema, que a coloração C' não troca a cor de nenhum vértice da coloração C, isto é, a coloração C' atribuirá cores somente aos vértices sem cor de forma que a coloração total seja convexa. Apresentamos a seguir, os resultados conhecidos na literatura para o r-RCGP.

Em 2007, Chor, Fellows, Ragan, Razgon, Rosamond e Snir [9] mostraram que o problema 1-RCGP é NP-difícil mesmo quando $|\mathcal{C}| = 2$. Para mostrar isto, apresentaram uma redução usando o problema 3-SAT (problema da satisfatibilidade para instâncias em que cada cláusula tem no máximo 3 literais), que sabemos ser NP-difícil [12]. Para este mesmo problema, obtiveram um algoritmo FPT de complexidade $O^*(8^k)$, onde o parâmetro k é a quantidade de vértices sem cor.

Esses autores mostraram que o problema 1-RCGP, para grafos com largura arbórea limitada por t, parametrizado por t, pertence à classe XP. Mostraram também que o problema 1-RCGP parametrizado pela largura arbórea do grafo é FPT.

Para $r \geq 2$, esses mesmos autores, mostraram que o r-RCGP, parametrizado por t, um limitante para a largura arbórea do grafo, é $W[1]$-difícil. A classe $W[1]$-difícil pode ser usada para mostrar que certos problemas são improváveis serem FPT.

Em 2008, Ponta, Hüffner e Niedermeier [20], melhoraram o algoritmo FPT para o problema 1-RCGP, abaixando a complexidade de $O^*(8^k)$ para $O^*(4^k)$.

Em 2008, Kammer e Tholey [13] obtiveram para o problema 1-RCGP um algoritmo de $(2 + \epsilon)$-aproximação para grafos com pesos e largura arbórea limitada.

3.2 Recoloração convexa de árvores

Nesta seção apresentamos os resultados conhecidos para o problema de recoloração convexa de árvores, que será denotado pela sigla **RCA**.

Seja (T, C) uma instância do RCA, que consiste em uma árvore T com n vértices, e uma coloração C. Um conjunto de cores $\mathcal{C}' \subseteq \mathcal{C}$ é definido como um **conjunto de cores ruins** se cada cor $c' \in \mathcal{C}'$ induz um subgrafo desconexo.

Em 2005, Moran e Snir [18] construíram um algoritmo para o RCA de complexidade $O(n\,\beta\,\Delta^{\beta})$, onde β é o número de cores ruins e Δ o grau máximo da árvore. Para um número fixo de cores, o algoritmo é polinomial no tamanho da entrada; logo este é um algoritmo FPT para o RCA. No mesmo artigo, esses autores apresentaram um algoritmo de programação dinâmica de complexidade $O((\beta/\log\beta)^{\beta})$. E por final, mostraram que se $|\mathcal{C}| = 2$, então, existe um algoritmo linear para o problema.

Em 2005, Moran e Snir [17] obtiveram um algoritmo de 3-aproximação, no modelo com pesos, que é quadrático no tamanho da entrada.

Em 2006, Bar-Yehuda, Feldman e Rawitz [3] melhoraram o resultado obtido por Moran e Snir [17] de 3-aproximação, exibindo uma $(2 + \epsilon)$-aproximação. Obtiveram também um algoritmo de programação dinâmica de complexidade $O^*((1/\epsilon)^2 4^{\epsilon})$.

Em 2007, Razgon [21] obteve um algoritmo de complexidade $O^*(256^k)$ para a seguinte versão parametrizada do RCA: o problema consiste em determinar se existe uma recoloração convexa que recolore no máximo k vértices.

Para esta mesma versão parametrizada do problema, em 2007, Bodlaender, Fellows, Lagnston, Ragan, Rosamond e Weyer [6] projetaram um algoritmo polinomial para obter uma kernelização para o RCA reduzindo uma instância de tamanho n para $O(k^2)$ vértices.

Em 2008, Kammer e Tholey [13] mostraram que o RCA restrito à classe de árvores coloridas em que cada cor aparece no máximo duas vezes é um problema NP-difícil. Para mostrar isto, eles fizeram uma redução utilizando o 3-SAT. Para esta classe de árvores, em 2009, Kanj e Kratsch [14] mostraram um algoritmo exato de complexidade $O^*(2^{0,293n})$.

Recentemente, Moran, Snir e Wing-Kin [19] obtiveram um algoritmo de programação dinâmica de complexidade $O^*(|\mathcal{C}|\,\Delta^{\alpha+2})$, onde α é o número de cores ruins, porém num sentido diferente do apresentado por Moran e Snir [18].

Em 2009, Kanj e Kratsch [14] obtiveram um algoritmo exato para o RCA de complexidade $O^*(2^{0,454n})$.

3.3 Recoloração convexa de caminhos

Mencionamos anteriormente que Moran e Snir [18] estudaram a complexidade de calcular a distância de recoloração em árvores, motivados por estudos filogenéticos. Nesse mesmo estudo, eles mostraram que no caso especial de caminhos (RCC) o problema ainda permanece NP-difícil, mesmo no modelo sem pesos. Para provar isto, reduziram o problema 3-SAT para o RCC.

Eles também mostraram que, mesmo para caminhos nos quais toda aresta tem seus extremos coloridos com cores distintas (os chamados **caminhos zebra**), o problema continua NP-difícil.

Moran e Snir [18] construíram um algoritmo de programação dinâmica para o RCC de complexidade $O(\beta\, n\, 2^{\beta})$, na sua versão com pesos, utilizando o conceito de cobertura convexa.

Por definição, um conjunto X de vértices é uma **cobertura convexa** para um caminho colorido (P, C) se a coloração parcial $C_X = C|_{[V \setminus X]}$ é convexa, isto é, C pode ser transformada em uma coloração convexa por trocar a cor dos vértices de X.

Em 2005, Moran e Snir [17] apresentaram um algoritmo de 2-aproximação para o RCC, na sua versão com pesos, de complexidade $O(|\mathcal{C}|n)$. No final do próximo capítulo apresentaremos esse algoritmo.

Lembramos que denotamos por **2-RCC** o caso especial do RCC em que a instância consiste em um caminho colorido, onde cada cor aparece no máximo duas vezes.

Em 2009, Kanj e Kratsch [14] mostraram que o 2-RCC é NP-difícil. A redução foi feita usando o problema do conjunto independente máximo. Esse resultado mostra que o RCC é NP-difícil. Trata-se de uma solução mais simples comparada à redução apresentada por Moran e Snir [18] para o problema RCC. Mostraremos este resultado na próxima seção.

Kanj e Kratsch [14] construíram algoritmos exatos para o 2-RCC de complexidade $O^*(2^{n/4})$, e para o RCC, de complexidade $O^*(2^{4n/9})$, ambos para o modelo sem pesos nos vértices.

3.3.1 Complexidade computacional

Mostraremos nesta seção o resultado obtido por Kanj e Kratsch [14] que estabelece que o problema 2-RCC é NP-difícil. Com isso, segue que o problema RCC é NP-difícil.

Lembramos que uma instância do problema 2-RCC consiste em um par (P, C), onde P é um caminho e $C : V \to \mathcal{C}$ é uma coloração, onde $\mathcal{C}$ denota o conjunto de cores utilizadas pela coloração C, e cada cor ocorre no máximo duas vezes em P.

Vamos mostrar que a versão de decisão do 2-RCC, denotado por 2-RCC-Decisão, é um problema NP-completo. Uma instância da versão de decisão consiste em um par $((P, C), \ell)$, onde (P, C) é uma instância do 2-RCC e ℓ é um inteiro não-negativo. O objetivo é decidir se existe uma recoloração convexa de P que troca a cor de no máximo ℓ vértices.

Kanj e Kratsch [14] provaram que 2-RCC-Decisão é NP-completo. Para isso, exibiram uma redução do problema do conjunto independente máximo (PCI) para este problema.

A versão de decisão do problema do conjunto independente máximo consiste em, dado um par (G, k), onde $G = (V, A)$ é um grafo e k é um número inteiro, decidir se existe um conjunto independente em G de cardinalidade k.

Teorema 3.3.2 (Kanj e Kratsch, 2009) *O problema* 2-RCC-Decisão *é* NP-*completo.*

Prova Seja $((P, C), \ell)$ uma instância do problema 2-RCC-Decisão. É fácil ver que 2-RCC-Decisão está em NP, pois dado um certificado de uma solução para este problema, conseguimos verificar em tempo polinomial no tamanho da instância se, de fato, este certificado está correto, ou seja, é uma recoloração convexa de P que troca a cor de no máximo ℓ vértices.

Vejamos agora como reduzir o problema PCI para o problema 2-RCC-Decisão.

Dada uma instância (G, k) do PCI, vamos construir uma instância $((P, C), \ell)$, de forma que, se temos um conjunto independente em G de cardinalidade pelo menos k, então temos uma recoloração convexa de (P, C) que faz no máximo ℓ trocas. O valor de ℓ, que depende de k, será indicado a seguir.

No que segue, $n(G) := |V(G)|$, $m(G) := |A(G)|$ e $P = (v_1, \ldots, v_n)$.

Descreveremos a construção formalmente, e também ilustraremos através de um exemplo como é feita esta construção. Considere o grafo G abaixo:

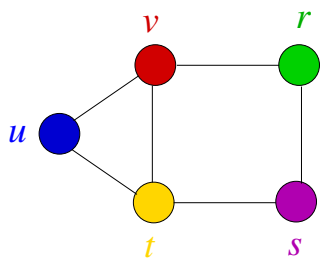

Figura 3.1

- Para cada vértice $v \in V(G)$, criamos dois vértices v_1 e v_2 em P, de mesma cor. O caminho P consistirá de n subcaminhos, que não se intersectam, cada um dos

quais é da forma $(v_1, \ldots, v_2)$, onde v é um vértice de $V(G)$. Estes subcaminhos são chamados de *subcaminhos-de-vértices*. Veremos a seguir quais são os demais vértices que ocorrerão em P. Se u e v são vértices distintos de G, então os vértices u_i e v_i devem ter cores distintas.

Para o caso do grafo G da Figura 3.1, onde $V(G) = \{u, v, r, s, t\}$, temos que P contém os subcaminhos-de-vértices $(u_1, \ldots, u_2)$, $(v_1, \ldots, v_2)$, $(r_1, \ldots, r_2)$, $(s_1, \ldots, s_2)$ e $(t_1, \ldots, t_2)$, sendo a ordem desses subcaminhos em P dada por uma ordem que fixamos arbitrariamente para os vértices de $V(G)$. Além disso, quaisquer dois subcaminhos-de-vértices não se intersectam. Assim, por ora, para o exemplo da figura, temos o seguinte:

$$P := (u_1, \ldots, u_2, \ldots v_1, \ldots, v_2, \ldots r_1, \ldots, r_2, \ldots s_1, \ldots, s_2, \ldots t_1, \ldots, t_2).$$

- Para cada aresta $e = uv \in A(G)$, criamos também dois vértices e_{uv}, e_{vu} em P, de mesma cor, porém com uma cor distinta das já existentes. O vértice e_{uv} fica no interior do subcaminho $(u_1, \ldots, u_2)$, e o vértice e_{vu} fica no interior do subcaminho $(v_1, \ldots, v_2)$, formando um *subcaminho-de-aresta* $(e_{uv}, \ldots, e_{vu})$.

  No exemplo em foco, obtemos

$$P := (u_1, e_{uv}, e_{ut}, u_2, \ldots v_1, e_{vu}, e_{vt}, e_{vr}, v_2, \ldots, r_1, e_{rv}, e_{rs}, r_2,$$

$$\ldots s_1, e_{sr}, e_{st}, s_2, \ldots t_1, e_{ts}, e_{tv}, e_{tu}, t_2).$$

- Entre quaisquer dois subcaminhos-de-vértices que são consecutivos em P, digamos $(u_1, \ldots, u_2)$ e $(v_1, \ldots, v_2)$, inserimos um número $N > m(G) + n(G) - k$ de vértices, que chamaremos de *subcaminho intermediário*, onde cada um dos vértices deste subcaminho tem uma cor distinta dos demais vértices de P. Observe que teremos um total de $(n(G) - 1) \times N$ vértices para este tipo de subcaminho. Para simplificar, escreveremos somente N_i para representar esse tipo de subcaminho de P, e consideraremos que N_i fica entre o i-ésimo e o $(i + 1)$-ésimo subcaminhos de vértices.

$$P := (u_1, e_{uv}, e_{ut}, u_2, N_1, v_1, e_{vu}, e_{vt}, e_{vr}, v_2, N_2, r_1, e_{rv}, e_{rs}, r_2,$$

$$N_3, s_1, e_{sr}, e_{st}, s_2, N_4, t_1, e_{ts}, e_{tv}, e_{tv}, e_{tu}, t_2).$$

Seja (P, C) o par construído acima, e seja $\ell := m(G) + n(G) - k$. Temos que $((P, C), \ell)$ é uma instância do problema 2-RCC-Decisão.

Vamos mostrar agora que a redução do PCI para 2-RCC-Decisão tem a propriedade desejada. Primeiramente, observamos que a construção de (P, C) pode ser feita em tempo polinomial no tamanho do grafo G.

- Suponha que G tenha um conjunto independente I de cardinalidade k. No exemplo em foco, tomemos $k = 2$, e $I = \{u, s\}$. Vamos descrever uma recoloração C_I de (P, C) que faz no máximo $\ell := m(G) + n(G) - k$ trocas.

  1. Para cada vértice $w \in I$, a recoloração C_I mantém a cor dos vértices w_1 e w_2 que são os extremos do subcaminho-de-vértices $(w_1, \ldots, w_2)$ em P e recolore com a cor de w_1 (que é igual a de w_2) todos os vértices que estão neste subcaminho.
  2. A recoloração C_I mantém a cor de cada vértice que está num subcaminho intermediário.
  3. Para cada vértice $w \in V(G) \setminus I$, a cor do primeiro vértice do subcaminho-de-vértice $(w_1, \ldots, w_2)$ é mantida e o último vértice desse subcaminho recebe a cor do seu predecessor.
  4. Para os vértices dos subcaminhos-de-arestas que não foram recoloridos no primeiro passo e que também não são um extremo de um subcaminho-de-aresta que teve o seu outro extremo recolorido no primeiro passo (é o caso do subcaminho $(e_{vt}, \ldots, e_{tv})$, no exemplo em foco), aplicamos a mesma ideia do terceiro passo: o primeiro vértice do subcaminho-de-aresta tem a sua cor mantida, e o último vértice recebe a cor do seu predecessor.

$$P := (u_1, e_{uv}, e_{ut}, u_2, N_1, v_1, e_{vu}, e_{vt}, e_{vr}, v_2, N_2, r_1, e_{rv}, e_{rs}, r_2,$$

$$N_3, s_1, e_{sr}, e_{st}, s_2, N_4, t_1, e_{ts}, e_{tv}, e_{tu}, t_2).$$

Afirmamos que a quantidade de trocas de cor feita pela recoloração C_I é no máximo $m(G) + n(G) - k$. De fato, como em G não existem arestas com extremos em quaisquer dois elementos de I, nenhum subcaminho-de-aresta tem seus dois extremos recoloridos em C_I. Portanto, para cada subcaminho-de-aresta trocamos

a cor de no máximo um dos seus extremos (assim, temos um total de no máximo $m(G)$ trocas desse tipo). Para cada subcaminho-de-vértice correspondente a um vértice em $V(G) \setminus I$, a recoloração C_I troca a cor de exatamente um dos seus extremos (lembramos que para os vértices em I, os extremos dos subcaminhos-de-vértices correspondentes têm a sua cor mantida por C_I). Assim, C_I faz um total de $n(G)-k$ trocas relativamente aos extremos dos subcaminhos-de-vértices. Portanto, a quantidade total de trocas feita pela recoloração C_I é no máximo $m(G)+n(G)-k$.

- Suponha que (P, C) admita uma recoloração convexa que troca a cor de no máximo $\ell := m(G) + n(G) - k$ vértices. Vamos mostrar que neste caso G tem um conjunto independente de cardinalidade k.

  Seja C' uma recoloração convexa que troca a cor de no máximo ℓ cores, e que mantém o maior número de cores da coloração C.

  Primeiramente, é fácil ver que C' troca a cor de pelo menos um dos extremos de cada subcaminho-de-aresta.

  De fato, isto ocorre porque cada subcaminho-de-aresta contém pelo menos um subcaminho intermediário que tem mais do que ℓ vértices. Além disso, os extremos de cada subcaminho-de-aresta são dois a dois distintos. Assim, cada aresta de G (e portanto pelo menos um dos extremos de um subcaminho-de-aresta) força pelo menos uma troca de cor em C'.

  Omitiremos aqui a prova do seguinte fato, que pode ser obtida analisando-se os possíveis casos em que alguma cor de C não ocorre mais em C', permitindo assim obter uma outra coloração que contradiz a escolha de C'.

  Fato: O conjunto de cores de C' é igual ao de C.

  Usando o fato acima e a observação feita anteriormente podemos concluir que C' troca a cor de exatamente um dos extremos de cada subcaminho-de-aresta. Além disso, concluímos que nos subcaminhos intermediários nenhum vértice é recolorido, já que cada vértice destes subcaminhos possui uma cor única.

  Como o número total de vértices recoloridos em C' é no máximo $m(G) + n(G) - k$, e há exatamente $m(G)$ trocas de cor correspondentes a um dos extremos de cada subcaminho-de-aresta (em P), então no máximo $n(G) - k$ outras trocas podem ser efetuadas. Estas devem ser trocas correspondentes aos $n(k)$ subcaminhos-de-vértices. Portanto, pelo menos k subcaminhos-de-vértices têm ambos os seus extremos com a cor inalterada. Seja I o conjunto dos vértices em G correspondentes a esses subcaminhos.

Como C' troca a cor de exatamente um dos extremos de cada subcaminho-de-aresta, nenhum subcaminho-de-aresta tem seus dois extremos situados em subcaminhos-de-vértices correspondentes aos vértices em I. Portanto, em G não existe nenhuma aresta com ambos os extremos em I, ou seja, I é um conjunto independente. Como I tem pelo menos k vértices, a prova está completa. □

Capítulo 4

Algoritmo de aproximação para o problema 2-RCC

Em 2005, Moran e Snir [17; 18] provaram que RCC, o problema da recoloração convexa de caminhos, é NP-difícil, e apresentaram um algoritmo de 2-aproximação. Em 2009, Kanj e Kratsch [14] mostraram que o 2-RCC, o caso especial de RCC em que as instâncias consistem em caminhos coloridos nos quais cada cor ocorre no máximo 2 vezes, continua NP-difícil.

Neste capítulo apresentamos um algoritmo de $\frac{3}{2}$-aproximação para o problema 2-RCC. Mostramos também que a razão de aproximação $\frac{3}{2}$ é justa. Kanj e Kratsch [14] desenvolveram algoritmos exatos para os problemas RCC e 2-RCC. Não encontramos na literatura algoritmos de aproximação para o 2-RCC.

O algoritmo de aproximação que descrevemos neste capítulo, chamado Aprox2RCC, foi inspirado no algoritmo de 2-aproximação obtido por Moran e Snir [17; 18] para o problema RCC. O algoritmo que obtivemos é simples de ser entendido e implementado. No entanto, a análise de desempenho deste algoritmo mostrou-se relativamente mais complicada comparada à análise do algoritmo de 2-aproximação.

Apresentaremos primeiramente o algoritmo Aprox2RCC que desenvolvemos para o 2-RCC. Depois disso, mostraremos o algoritmo de 2-aproximação para o RCC de Moran e Snir, que poderá ser facilmente entendido, uma vez que os conceitos principais envolvidos são os mesmos.

4.1 Preliminares

Demonstraremos primeiramente alguns lemas que serão usados na análise do algoritmo Aprox2RCC a ser descrito mais adiante.

Para simplificar, convencionamos que — neste capítulo — quando escrevemos caminho colorido (P, C), estamos supondo que (P, C) refere-se a uma instância do problema 2-RCC. Isto é, no caminho colorido P, cada cor ocorre no máximo 2 vezes. A menos de

menção em contrário, também supomos que $P = (v_1, \ldots, v_n)$ e $\mathcal{C}$ é o conjunto de cores usadas pela coloração C.

Se (P, C') é uma recoloração convexa de (P, C), então lembramos que custo(C') denota o número de vértices v em P para os quais $C(v) \neq C'(v)$, ou seja, custo(C') é precisamente o número de trocas de cor efetuadas por C'. Lembramos também que opt(P, C) denota o custo de uma solução ótima para a instância (P, C) do problema em foco (que pode ser 2-RCC ou RCC).

O seguinte lema tem um papel importante no algoritmo Aprox2RCC. Um dos passos desse algoritmo é baseado neste resultado.

Lema 4.1.1 *Seja (P, C) uma instância do problema* 2-RCC, *com $P = (v_1, \ldots, v_n)$, e sejam v_i e v_{i+1} vértices em P, tais que $C(v_i) = C(v_{i+1}) = d$. Então, para toda recoloração convexa ótima C^* de (P, C), temos que $C^*(v_i) = C^*(v_{i+1}) = d$.*

Prova É imediato que se $i = 1$ ou $i = n - 1$, o resultado vale. Suponhamos então que $1 < i < n - 1$. Seja C^* uma recoloração convexa ótima de (P, C). Suponha primeiro que C^* troque a cor de exatamente um dos vértices v_i, v_{i+1}; digamos que seja a de v_i.

Claro que, $C^*(v_{i-1}) \neq C^*(v_{i+2})$, caso contrário a recoloração C^* não seria convexa. Seja R uma recoloração de (P, C) tal que $R(v) = C^*(v)$ para todo $v \neq v_i$ e $R(v_i) = C(v_i) = d$. É fácil ver que R é uma recoloração convexa, já que $C^*(v)$ é uma coloração convexa e $R(v_i) = R(v_{i+1}) = d$. Além disso, a recoloração R realiza uma troca a menos que a recoloração C^*, o que contraria o fato de C^* ser uma recoloração convexa ótima. Por simetria, concluímos que C^* não troca apenas a cor de v_{i+1}.

Suponhamos agora que C^* troque as cores de ambos os vértices v_i e v_{i+1}. Neste caso, podemos supor que C^* não atribui a cor d a nenhum vértice v_j com $j < i$ ou $j > i + 1$. De fato, se isso ocorresse, poderíamos tomar uma outra coloração, digamos $\widehat{C}$, que é igual a C^*, a menos do fato de que os vértices com a cor d em C^* recebem em $\widehat{C}$ uma outra cor distinta de d. Neste caso, custo($\widehat{C}$) $\leq$ custo(C^*), já que cada ocorrência da cor d em C^* implica que houve uma troca de cor em C^*, e portanto ao substituir a cor d por uma outra não vai tornar a coloração $\widehat{C}$ mais onerosa do que C^*. (A contradição que vamos obter em seguida poderia ser obtida com $\widehat{C}$ no lugar de C^*.)

Vamos analisar separadamente os seguintes dois casos: $C^*(v_i) \neq C^*(v_{i+1})$ e $C^*(v_i) = C^*(v_{i+1})$.

Caso (a) $C^*(v_i) \neq C^*(v_{i+1})$.

Neste caso, vale que $C^*(v_{i-1}) \neq C^*(v_{i+2})$, já que C^* é convexa. Seja R uma recoloração de (P, C) tal que $R(v) = C^*(v)$ para todo $v \notin \{v_i, v_{i+1}\}$ e $R(v_i) = R(v_{i+1}) = d$. Claramente, R é uma recoloração convexa de (P, C). Além disso,

R faz duas trocas a menos que a recoloração C^*, contrariando a otimalidade de C^*.

Caso (b) $C^*(v_i) = C^*(v_{i+1})$.

Neste caso, podemos ter $C^*(v_{i-1}) \neq C^*(v_{i+2})$ ou $C^*(v_{i-1}) = C^*(v_{i+2})$. Para o caso em que $C^*(v_{i-1}) \neq C^*(v_{i+2})$, tomando a recoloração R definida no caso (a), novamente contrariamos a otimalidade de C^*.

Para finalizar, considere o caso em que $C^*(v_{i-1}) = C^*(v_{i+2}) := f$.

Seja v_j o primeiro vértice em P tal que $C^*(v_j) = f$; e seja v_k o último vértice em P tal que $C^*(v_k) = f$. Como a cor f ocorre no máximo 2 vezes em C, então (pelo menos) um dos intervalos $I_1 := [j, i-1]$ ou $I_2 := [i+2, k]$ contém no máximo um índice p tal que $C(v_p) = f$.

Caso (b.1) Se for o intervalo I_1, considere a recoloração R' assim definida: $R'(v_t) = C^*(v_t)$ para $1 \leq t < j$, $R'(v_t) = d$ para $j \leq t \leq i+1$, e $R'(v_t) = C^*(v_t)$ para $i+1 < t \leq n$. Claramente, R' é uma recoloração convexa de (P, C). Além disso, R' faz menos trocas que a recoloração C^*. Portanto, R' contraria a otimalidade da recoloração C^*.

Caso (b.2) Se for o intervalo I_2, assim como no caso anterior, podemos facilmente construir a partir de C^*, uma recoloração que contraria a otimalidade de C^*.

Concluímos, portanto, que não existe uma recoloração convexa ótima que recolore ambos os vértices v_i, v_{i+1}. Com isso, completamos a prova do lema. □

Seja (P, C) uma instância do 2-RCC, onde $P = (v_1, \ldots, v_n)$. Dizemos que um vértice v_i em (P, C) é **colorido fracamente** se:

(a) v_i é vértice interno de P;

(b) não existe em P nenhum outro vértice com a cor de v_i;

(c) ambos os vizinhos de v_i têm a mesma cor, isto é, $C(v_{i-1}) = C(v_{i+1})$.

No algoritmo Aprox2RCC os vértices coloridos fracamente são tratados de uma forma diferenciada. Basicamente, estes são removidos do caminho P, conforme explicamos a seguir.

Suponha que (P, C) tem um vértice v_i colorido fracamente. Seja (P', C') o par obtido de (P, C), assim construído:

- P' é o caminho obtido de P removendo-se o vértice v_i e acrescentando-se a aresta $\{v_{i-1}, v_{i+1}\}$;
- a coloração C' de P' é a restrição de C a P', isto é, $C' = C|_{P'}$.

Este procedimento cria em P' um (novo) par de vértices adjacentes que têm a mesma cor. Dizemos que um tal par (v_{i-1}, v_{i+1}) de vértices de mesma cor é um **par forte** em P (e em P'). Se (P', C') tem outros vértices coloridos fracamente, repetimos esse procedimento para cada um destes vértices até obtermos um caminho sem vértices coloridos fracamente. Obtemos assim um novo caminho que chamamos de **caminho forte**.

Denominamos de **fase de limpeza** a sequência de operações realizadas para se obter a partir do par original (P, C) um novo par (P', C'), onde P' é um caminho forte.

Na Figura 4.1 ilustramos a construção de um caminho forte.

(P, C) : **A B A E F H E G I G L L M H M N R N R**

$\Downarrow$ identificamos os vértices coloridos fracamente

A B A E F H E G I G L L M H M N R N R

$\Downarrow$ removemos todos eles

(P', C') : **A A E F H E G G L L M H M N R N R**

(P', C') é um **caminho forte**

Figura 4.1: *Ilustração da fase de limpeza.*

O resultado a seguir mostra uma forte relação que existe entre $\text{opt}(P, C)$ e $\text{opt}(P', C')$.

Lema 4.1.2 *Seja (P, C) uma instância do problema* 2-RCC *que tem k vértices coloridos fracamente. Seja (P', C') o par obtido de (P, C) após a fase de limpeza. Então* $\text{opt}(P, C) = \text{opt}(P', C') + k$.

Prova Primeiramente observamos que é imediato que $\text{opt}(P, C) \leq \text{opt}(P', C') + k$. Para isto, basta notar que uma solução viável para a instância (P, C) pode ser obtida da

seguinte forma: toma-se uma solução ótima, digamos (P', C^*) para a instância (P', C'), a qual sabemos, pelo Lema 4.1.1, mantém inalterada a cor original dos pares fortes. Assim, tomando-se essa solução (P', C^*), podemos obter uma solução viável para a instância (P, C) estendendo-se a coloração C^* a P, simplesmente atribuindo-se, a cada vértice colorido fracamente, a cor do par forte correspondente. Claramente, essa solução viável para a instância (P, C) faz $\text{opt}(P', C') + k$ trocas de cor. Assim, $\text{opt}(P, C) \leq \text{opt}(P', C') + k$.

Vamos agora provar que $\text{opt}(P, C) \geq \text{opt}(P', C') + k$. Seja (P, C^*) uma solução ótima para a instância (P, C) do problema 2-RCC que mantém o maior número possível de pares fortes com sua cor original.

Suponha que C^* troque a cor de um par forte (v_{i-1}, v_{i+1}) de P. Suponha que $C(v_{i-1}) = C(v_{i+1}) = d$. Vamos analisar dois casos possíveis.

Caso (a) A recoloração C^* troca as cores de v_{i-1} e de v_{i+1}.

> Podemos supor, sem perda de generalidade, que C^* não atribui a cor d a nenhum vértice de P (se houver em C^* vértices com a cor d estes podem receber uma outra cor, sem aumentar o custo da recoloração obtida, e sem diminuir o número de pares fortes que continuam com a cor original).
>
> Seja $\widehat{C}$ a coloração assim definida: $\widehat{C}(v_{i-1}) = \widehat{C}(v_i) = \widehat{C}(v_{i+1}) = d$ e $\widehat{C}(v) = C^*(v)$ para $v \notin \{v_{i-1}, v_i, v_{i+1}\}$. Claramente, $\widehat{C}$ é uma recoloração convexa de (P, C), e faz menos trocas do que a recoloração C^*. Mas isto contraria a otimalidade de C^*, e portanto este caso não ocorre.

Caso (b) A recoloração C^* troca apenas a cor de um dentre v_{i-1} e v_{i+1}.

> Suponha que C^* troca a cor de v_{i-1}. Seja $\widetilde{C}$ a coloração como definimos abaixo: $\widetilde{C}(v_{i-1}) = \widetilde{C}(v_i) = d$ e $\widetilde{C}(v) = C^*(v)$ para $v \notin \{v_{i-1}, v_i\}$. Note que, $C^*(v_{i+1}) = d$. Claramente $\widetilde{C}$ é uma recoloração convexa de (P, C) e $\text{custo}(\widetilde{C}) \leq \text{custo}(C^*)$. Como C^* é uma recoloração ótima, concluímos que $\widetilde{C}$ e C^* têm o mesmo custo. Mas como $\widetilde{C}$ mantém a cor do par forte (v_{i-1}, v_{i+1}), temos uma contradição à escolha de C^*.

Concluímos então que C^* mantém todos os pares fortes de P com a cor original. Portanto, C^* necessariamente troca a cor de cada vértice colorido fracamente atribuindo-lhe a cor do par forte correspondente.

Considere agora a coloração $\widetilde{C}$ que é a restrição de C^* aos vértices de P'. Claramente $\widetilde{C}$ é uma recoloração convexa de (P', C'). Logo, $\text{custo}(\widetilde{C}) \geq \text{opt}(P', C')$. Como $\text{custo}(\widetilde{C}) = \text{opt}(P, C) - k$, temos que $\text{opt}(P, C) \geq \text{opt}(P', C') + k$, como queríamos demonstrar. □

Os lemas 4.1.1 e 4.1.2 sugerem a concepção de um algoritmo para o 2-RCC baseado num algoritmo para o caso especial em que os caminhos são fortes. Veremos como fazer

isso na próxima seção.

4.2 Algoritmo para o problema 2-RCC-Forte

Para facilitar, chamamos de 2-RCC-Forte o problema 2-RCC restrito a caminhos fortes.

Veremos em seguida um algoritmo de aproximação para o problema 2-RCC-Forte. Antes porém, precisamos introduzir algumas definições e a notação que será usada.

Para um par (P, C), e uma cor $d \in \mathcal{C}$, seja

$$C[d] := \{v \in V(P) : \ C(v) = d\}.$$

O algoritmo que veremos baseia-se fortemente no seguinte conceito. Se Q é um subcaminho de P, definimos a **multa de** d **sobre** Q**, com respeito a** C, como

$$\text{multa}_{d,C}(Q) = \frac{1}{2}|V(Q) \cap \overline{C[d]}| + |\overline{V(Q)} \cap C[d]|.$$

Ou seja, $\text{multa}_{d,C}(Q)$ é calculada somando-se $\frac{1}{2}$ para cada vértice em Q que não possui a cor d, e somando-se 1 para cada vértice de cor d que não está em Q. Se $Q = \emptyset$, então $\text{multa}_{d,C}(Q) = |C[d]|$. Intuitivamente, um subcaminho sobre o qual a multa de uma cor d é baixa, é um forte candidato a receber a cor d na recoloração procurada. Isso motiva o conceito de multa mínima de uma cor, que definimos logo a seguir.

Observamos que a multa de uma cor d sobre um subcaminho é sempre relativa à uma certa coloração. Quando esta coloração, digamos C, está clara pelo contexto, em vez de escrevermos $\text{multa}_{d,C}(Q)$, escrevemos simplesmente $\text{multa}_d(Q)$.

Exibimos na Figura 4.2 alguns exemplos de multas.

Para cada cor d, supondo que a coloração C esteja clara pelo contexto, denotamos por m^*_d a **multa mínima de** d assim definida:

$$m^*_d = \min\{\text{multa}_d(Q) : Q \text{ é um subcaminho de } P\}.$$

Observe que para qualquer cor d, temos que $m^*_d \in \{0, \frac{1}{2}, 1\}$. Temos que $m^*_d = 0$ quando Q é formado por dois vértices adjacentes de cor d e quando Q é formado por um vértice de cor d, tal que este vértice é a única ocorrência da cor d em P; $m^*_d = 1/2$ quando Q é um subcaminho de 3 vértices cujos extremos têm a cor d; $m^*_d = 1$ quando a distância entre as duas ocorrências da cor d é maior que 2.

Somente quando $m^*_d = 1$ podemos ter mais de um subcaminho Q tal que $m^*_d(Q) = 1$. Para este caso estabelecemos uma regra de desempate que veremos abaixo.

$$(P,C): A\ A\ E\ F\ H\ E\ G\ G\ L\ L\ \overbrace{M H M}^{Q}\ N\ R\ N\ R$$

$$\text{multa}_M(Q) = \tfrac{1}{2}.1 + 1.0 = \tfrac{1}{2}$$

$$(P,C): A\ A\ E\ F\ H\ E\ G\ G\ L\ L\ \overbrace{M}^{Q'}\ H\ M\ N\ R\ N\ R$$

$$\text{multa}_M(Q') = \tfrac{1}{2}.0 + 1.1 = 1$$

Figura 4.2: *Cálculo de multas para dois subcaminhos distintos.*

Para cada cor d, estamos interessados num único subcaminho B_d para o qual multa$(B_d) = m^*_d$. Convencionamos então o seguinte critério de desempate para definir B_d de maneira única. Sejam v_i e v_j vértices de cor d em P, onde $j \geq i + 3$. Se $C(v_{i-1}) \neq C(v_{i+1})$ então $B_d := (v_i)$, caso contrário, $B_d := (v_j)$. Apenas quando uma cor ocorre duas vezes em P, em vértices que estão à uma distância maior que 1, pode ocorrer empate. Nos demais casos, não ocorre empate.

Na Figura 4.2, a multa mínima da cor H ocorre quando Q é o subcaminho que tem apenas o primeiro vértice de cor H, e também quando Q é o subcaminho que tem apenas o segundo vértice de cor H.

Para cada cor d, o subcaminho B_d é chamado de **bloco**. Dizemos que um vértice em P é **livre** se este não pertence a nenhum bloco.

Adiante, definimos formalmente como cada bloco B_d é encontrado (veja o algoritmo AchaBloco).

Antes de prosseguir, mencionamos algumas observações que seguem das definições anteriores, e que serão úteis para o entendimento do próximo resultado.

- Para cada cor d, o bloco B_d começa num vértice de cor d e tem no máximo 3 vértices.
- O primeiro vértice do caminho P, digamos de cor d, sempre pertence ao bloco B_d. (Note que se $m^*_d = 1$ então $B_d = (v_1)$, pela regra de desempate.)
- Se um bloco $B_d = (v)$ está contido num bloco B_f, então v é o vértice do meio do bloco B_f, e há um outro bloco B_e, que ocorre antes de B_f, e que contém um vértice de cor d no meio.

- Cada vértice pertence a no máximo 2 blocos.
- Quando dois blocos B_d e B_f se intersectam em mais do que um vértice, então esses blocos têm as cores (d, f, d) e (f, d, f).

O algoritmo de aproximação AproxRCC-Forte

Descrevemos a seguir o algoritmo AproxRCC-Forte para o problema 2-RCC-Forte. Este algoritmo por sua vez, utiliza uma subrotina que chamamos de AchaBloco, que encontra, para uma cor d, o bloco B_d. Sua descrição é apresentada em seguida.

Em linhas gerais, o algoritmo AproxRCC-Forte primeiramente encontra, para cor d, o bloco B_d. A ideia básica é que cada bloco B_d ajudará a definir a cor que seus vértices receberão. Quando um bloco B_d não intersecta nenhum outro, então os vértices em B_d recebem a cor d. Quando um vértice v pertence a mais do que um bloco, então v recebe a cor do bloco que o contém e que começa antes. Para implementar essa ideia, o caminho P é percorrido a partir de seu início: se é detectado que um vértice v pertence a um bloco B_d então v e todos os demais vértices do bloco B_d recebem a cor d (não importando se B_d intersecta outro bloco). Quando se chega ao fim desse bloco B_d, o próximo vértice w ou é livre ou está num outro bloco, digamos B_f (que é o único bloco que contém w). Se w é livre, então w recebe a cor do seu predecessor, caso contrário, w recebe a cor f. (Observamos que essa última situação ocorre no passo 6 do algoritmo AproxRCC-Forte.)

Algoritmo **AproxRCC-Forte**

Entrada: Um par (P, C), onde P é um caminho forte, $P = (v_1, \ldots, v_n)$, e o conjunto de cores de C é $\mathcal{C}$.

Saída: $(P, \widehat{C})$, uma recoloração convexa de (P, C).

Subrotina: AchaBloco encontra para uma cor d o bloco B_d.

1 **para cada** $d \in \mathcal{C}$, seja $B_d \leftarrow$ AchaBloco$((P, C), d)$
2 $d \leftarrow C(v_1)$
3 **para** $i = 1$ **até** n **faça**
4 **se** $v_i \in B_d$ **ou** v_i é livre
5 **então** $\widehat{C}(v_i) \leftarrow d$
6 **senão** seja k tal que $v_i \in B_k$
7 $\widehat{C}(v_i) \leftarrow k$; $d \leftarrow k$
8 **devolva** $(P, \widehat{C})$

Algoritmo **AchaBloco**

Entrada: Um par (P, C) e uma cor d.

Saída: Bloco B_d em P (sobre o qual a multa da cor d é mínima)

1 seja i o menor índice tal que $C(v_i) = d$.
2 **se** existe $j > i$ tal que $C(v_j) = d$
3 **então se** $j \leq i + 2$
4 **então** $B_d = \{v_i, \ldots, v_j\}$
5 **senão se** $C(v_{i-1}) \neq C(v_{i+1})$
6 **então** $B_d = \{v_i\}$
7 **senão** $B_d = \{v_j\}$
8 **senão** $B_d = \{v_i\}$
9 **devolva** B_d

Vale observar que o algoritmo AchaBloco pode ser implementado de modo que o caminho P seja percorrido uma única vez para descobrir, para toda cor d, o bloco B_d correspondente. Ou seja, não é preciso percorrer P cada vez que uma nova cor d é

considerada (conforme se deduz da descrição de AchaBloco).

Para um melhor entendimento do algoritmo AproxRCC-Forte, exibimos na Figura 4.3 uma instância para o problema 2-RCC-forte e a solução devolvida pelo algoritmo AproxRCC-Forte.

$$(P, C): \; A\,A\,E\,F\,H\,E\,G\,G\,L\,L\,M\,H\,M\,N\,R\,N\,R$$

$\Downarrow$ Para cada cor d, encontramos B_d (execução do passo 1)

$$(P, C): \; \overbrace{A\,A}^{B_A}\;\overbrace{E}^{B_E}\;\overbrace{F}^{B_F}\;\overbrace{H}^{B_H}\;E\;\overbrace{G\,G}^{B_G}\;\overbrace{L\,L}^{B_L}\;\overbrace{M\,H\,M}^{B_M}\;\overbrace{N\;\underbrace{R\,N}_{B_R}}^{B_N}\;R$$

$\Downarrow$ Execução dos passos 2 a 8

$$(P, \widehat{C}): \; A\,A\,E\,F\,H\,H\,G\,G\,L\,L\,M\,M\,M\,N\,N\,N\,R$$

Figura 4.3: *Execução do algoritmo AproxRCC-Forte.*

Análise de desempenho do algoritmo AproxRCC-Forte

Para simplificar, usaremos a seguinte notação condensada na prova do próximo teorema. Se $C(v_i) = d_i$, para $j \leq i \leq k$, podemos expressar isso escrevendo $C(v_j, v_{j+1}, \ldots, v_k) = (d_j, d_{j+1}, \ldots, d_k)$.

Teorema 4.2.1 *O algoritmo* AproxRCC-Forte *é uma* 3/2*-aproximação para* 2-RCC-Forte. *Além disso, a razão* 3/2 *é justa.*

Prova É fácil ver que a recoloração $\widehat{C}$ devolvida pelo algoritmo AproxRCC-Forte é convexa. Isto segue do fato de que o algoritmo percorre o caminho P a partir do seu início, e sempre que começa a atribuir uma nova cor, digamos d, esta cor é atribuída sequencialmente a todos os vértices do bloco B_d e aos vértices livres que seguem este bloco, se existirem; e depois disso, essa cor d não é atribuída novamente, pois para cada cor d existe um único bloco B_d.

Vamos provar que

$$\text{custo}(\widehat{C}) \leq \frac{3}{2}\text{custo}(\widetilde{C}),$$

onde $(P, \widetilde{C})$ é uma recoloração convexa qualquer de (P, C).

Primeiramente, vamos mostrar que

$$\text{custo}(\widehat{C}) \leq \sum_{d \in \mathcal{C}} m_d^*,$$

onde m_d^* é a multa mínima da cor d, com respeito a C.
Para simplificar, chame de S^* a soma $\sum_{d \in \mathcal{C}} m_d^*$.
Observe que

$$\text{custo}(\widehat{C}) = \sum_{d \in \mathcal{C}} |\{v \in V(P) : C(v) = d \ \text{ e } \ \widehat{C}(v) = d', \, d' \neq d\}|.$$

Para provar que custo$(\widehat{C}) \leq \sum_{d \in \mathcal{C}} m_d^*$, vamos mostrar que, se uma cor d sofre $t_d > 0$ trocas, então o custo t_d é "pago" por um conjunto de uma ou mais multas mínimas, cuja soma é pelo menos t_d. Além disso, cada multa mínima é usada para pagar no máximo uma troca.

Seja d uma cor que é trocada pelo algoritmo. Suponha que $C(v_i) = d$ e $\widehat{C}(v_i) = d'$. Neste caso, v_i é livre ou $v_i \in B_{d'}$.

Antes de analisarmos cada caso, considere a Figura 4.4, onde exibimos todos os tipos possíveis de blocos, e o comportamento do algoritmo AproxRCC-Forte.

$$(P, C) : \overbrace{A\,D\,A}^{B_A} \ \overbrace{E}^{B_E} \ \overbrace{F}^{B_F} \ \overbrace{H}^{B_H} \ E \ \overbrace{G\,G}^{B_G} \ \overbrace{L \ \underbrace{D}_{B_D} \ L}^{B_L} \ \overbrace{M\,H\,M}^{B_M} \ \overbrace{N \ \underbrace{R\,N}_{B_R} \ R}^{B_N}$$

$\Downarrow$ Após execução dos passos 2 a 8

$$(P, \widehat{C}) : A\,A\,A\,E\,F\,H\,H\,G\,G\,L\,L\,L\,M\,M\,M\,N\,N\,N\,R$$

$$\text{custo}(\widehat{C}) = 5$$

$$\begin{aligned}\sum_{d \in \mathcal{C}} m_d^* &= m_A^* + m_D^* + m_E^* + m_F^* + m_H^* + m_G^* + m_L^* + m_M^* + m_N^* + m_R^* \\ &= 1/2 + 1 + 1 + 0 + 1 + 0 + 1/2 + 1/2 + 1/2 + 1/2 \\ &= 11/2\end{aligned}$$

Figura 4.4: *Tipos de blocos de multa mínima.*

- Se v_i é livre, então $B_d = (v_j)$, onde $j < i - 2$. (Na Figura 4.4, podemos tomar como v_i o segundo vértice com a cor E, e neste caso, v_j é o primeiro vértice com a cor

E). Note que, o vértice v_j não tem a sua cor trocada pelo algoritmo. Assim, $t_d = 1$ e em S^* temos a multa $m^*_d = 1$ que paga t_d.

- Se $v_i \in B_{d'}$, então $B_{d'} = (v_{i-1}, v_i, v_{i+1})$ e $C(v_{i-1}) = C(v_{i+1}) = d'$. Analisamos a seguir os possíveis casos para B_d.

  - $B_d = (v_i)$. Neste caso, existe v_j com $j < i - 2$ tal que $C(v_{j-1}, v_j, v_{j+1}) = (f, d, f)$, para alguma cor f distinta de d'. Então o algoritmo troca a cor de v_j (além de trocar a cor de v_i) e portanto, $t_d = 2$. Por outro lado, $m^*_d = 1$ e $m^*_{d'} = m^*_f = 1/2$. Como os vértices com as cores f e d' em C continuam com essas cores em $\widehat{C}$ (isto é $t_f = t_{d'} = 0$), podemos pagar t_d com a soma $m^*_d + m^*_f + m_{d'} = 2$. (Na Figura 4.4, podemos tomar como v_i o segundo vértice com a cor D, e neste caso, v_j é o primeiro vértice com a cor D.)

  - $B_d = (v_j)$.

    Caso (a). Se $j < i - 2$, então $C(v_{j-1}) \neq C(v_{j+1})$. Logo, $\widehat{C}(v_j) = C(v_j)$, ou seja, a cor d de v_j é mantida, e portanto $t_d = 1$. Como $m^*_d = 1$, essa multa paga por t_d. (Na Figura 4.4, podemos tomar como v_i o segundo vértice com a cor H, e neste caso, v_j é o primeiro vértice com a cor H.)

    Caso (b). Se $j > i+2$ temos duas possibilidades: $C(v_{j-1}) = C(v_{j+1})$ ou $C(v_{j-1}) \neq C(v_{j+1})$. No primeiro caso, $C(v_{j-1}, v_j, v_{j+1}) = (f, d, f)$, para alguma cor f distinta de d'. Como $C(v_{i-1}, v_i, v_{i+1}) = (d', d, d')$, então $t_d = 2$. Como os vértices com as cores f e d', em C, continuam com essas cores em $\widehat{C}$ (isto é, $t_f = t_{d'} = 0$), podemos usar $m^*_{d'}$ e m^*_f para pagar por t_d. Assim, a parcela $m^*_d + m^*_{d'} + m^*_f = 1 + 1/2 + 1/2 = 2$ paga por t_d.

    No segundo caso, isto é, quando $C(v_{j-1}) \neq C(v_{j+1})$, a segunda ocorrência da cor d é mantida, e portanto o algoritmo faz apenas a troca da primeira ocorrência da cor d. Neste caso, como $m^*_d = 1$, essa troca está computada em S^*.

  - $B_d = (v_i, v_{i+1}, v_{i+2})$.

    Neste caso, temos que $C(v_{i-1}, v_i, v_{i+1}, v_{i+2}) = (d', d, d', d)$, ou seja, os blocos $B_{d'}$ e B_d se intersectam nos vértices v_i e v_{i+1}. Então $t_d = 1$ (só a primeira ocorrência de d é alterada) e a cor d' que ocorre em C é mantida. Como $m^*_d + m^*_{d'} = 1/2 + 1/2 = 1$, essa soma paga por t_d. (Na Figura 4.4, tomar v_i como o primeiro vértice de cor R

Com isso, provamos que

$$\text{custo}(\widehat{C}) \leq \sum_{d \in \mathcal{C}} m^*_d.$$

Agora vamos mostrar que $\sum_{d\in\mathcal{C}} m_d^* \leq \frac{3}{2}\text{custo}(\widetilde{C})$, para qualquer recoloração convexa $(P, \widetilde{C})$ de (P, C).

Para cada cor $d \in \mathcal{C}$, denote por $\widetilde{Q}_d$ o subcaminho de P formado pelos vértices de cor d em $\widetilde{C}$. Note que, $\widetilde{Q}_d \cap \widetilde{Q}_{d'} = \emptyset$ para quaisquer cores distintas d e d', já que $\widetilde{C}$ é uma coloração convexa.

Assim, temos que

$$\sum_{d\in\mathcal{C}} m_d^* \leq \sum_{d\in\mathcal{C}} \text{multa}_d(\widetilde{Q}_d),$$

onde as multas são relativas à coloração C.

Por outro lado,

$$\begin{aligned}\sum_{d\in\mathcal{C}} \text{multa}_d(\widetilde{Q}_d) &= \sum_{d\in\mathcal{C}} \frac{1}{2}\left|\{v_i \in V(P) : \widetilde{C}(v_i) = d \text{ e } C(v_i) \neq d\}\right| \\ &\qquad + \left|\{v_i \in V(P) : \widetilde{C}(v_i) \neq d \text{ e } C(v_i) = d\}\right| \\ &= (3/2)\left|\{v_i \in V(P) : \widetilde{C}(v_i) \neq C(v_i)\}\right| \\ &= (3/2)\text{custo}(\widetilde{C}).\end{aligned}$$

Portanto, para qualquer recoloração convexa $(P, \widetilde{C})$ de (P, C), temos que

$$\text{custo}(\widehat{C}) \leq \sum_{d\in\mathcal{C}} m_d^* \leq \sum_{d\in\mathcal{C}} \text{multa}_d(\widetilde{Q}_d) = \frac{3}{2}\text{custo}(\widetilde{C}).$$

Isto vale, em particular, para uma recoloração convexa ótima. Logo, temos que

$$\text{custo}(\widehat{C}) \leq \frac{3}{2}\text{opt}(P, C).$$

Como o algoritmo AproxRCC-Forte é linear no comprimento do caminho, segue que ele é uma $\frac{3}{2}$-aproximação (polinomial) para o 2-RCC-Forte.

Resta agora mostrar que a razão de aproximação 3/2 é justa. Para isso, considere a seguinte instância (P, C), ilustrada na Figura 4.5.

Note que, para cada subcaminho indicado na Figura 4.5, o algoritmo AproxRCC-Forte faz 3 mudanças de cor, e com isso devolve uma recoloração $\widehat{C}$ tal que $\text{custo}(\widehat{C}) = 3k$. Por outro lado, uma recoloração ótima C^* é tal que $\text{custo}(C^*) = 2k$, conforme ilustramos na Figura 4.5. Isso mostra que a razão 3/2 do algoritmo AproxRCC-Forte é justa.

Com isso, concluímos a prova do teorema. □

$$\underbrace{\overbrace{a_1}^{B_{a_1}}\ \overbrace{b_1}^{B_{b_1}}\ \overbrace{c_1}^{B_{c_1}}\ a_1\ \overbrace{d_1\ d_1}^{B_{d_1}}\ b_1\ c_1}_{\textit{subcaminho } 1}\ \underbrace{\quad}_{\textit{subcaminho } 2}\ \ldots\ \underbrace{\overbrace{a_k}^{B_{a_k}}\ \overbrace{b_k}^{B_{b_k}}\ \overbrace{c_k}^{B_{c_k}}\ a_k\ \overbrace{d_k\ d_k}^{B_{d_k}}\ b_k\ c_k}_{\textit{subcaminho } k}$$

Instância (P, C)

$$(P, \widehat{C}):\ \ a_1\ b_1\ c_1\ c_1\ d_1\ d_1\ d_1\ d_1\ \ldots\ a_k\ b_k\ c_k\ c_k\ d_k\ d_k\ d_k\ d_k$$
$$(P, C^*):\ \ a_1\ a_1\ a_1\ a_1\ d_1\ d_1\ b_1\ c_1\ \ldots\ a_k\ a_k\ a_k\ a_k\ d_k\ d_k\ b_k\ c_k$$

Figura 4.5: *A razão 3/2 do AproxRCC-Forte é justa.*

4.2.2 Algoritmo de aproximação para o problema 2-RCC

Descrevemos a seguir o algoritmo Aprox2RCC, para o problema 2-RCC, que faz uso do algoritmo AproxRCC-Forte visto na seção anterior.

Algoritmo **Aprox2RCC**

Entrada: (P, C), uma instância do 2-RCC

Saída: $(P, \widetilde{C})$, uma recoloração convexa de (P, C)

Subrotina: AproxRCC-Forte para o problema 2-RCC-Forte

1 Seja (P', C') o caminho forte obtido de (P, C) `/*fase de limpeza*/`

2 $(P', \widehat{C}) \leftarrow$ AproxRCC-Forte (P', C')

3 **para cada** v_i em P **faça**

4 **se** $v_i \notin V(P')$

5 **então** $\widetilde{C}(v_i) \leftarrow \widehat{C}(v_{i-1})$

6 **senão** $\widetilde{C}(v_i) \leftarrow \widehat{C}(v_i)$

7 **devolva** $(P, \widetilde{C})$

Note que, nos passos 3 a 5, o algoritmo acima estende a P a coloração obtida para o caminho forte P' (conforme ideia usada na prova do Lema 4.1.2).

Usando o Lema 4.1.2, mostramos a seguir que o algoritmo Aprox2RCC herda a razão de aproximação do algoritmo AproxRCC-Forte.

Lema 4.2.3 *Se o algoritmo* AproxRCC-Forte *é uma α-aproximação para o problema* 2-RCC-Forte, *então o algoritmo* Aprox2RCC *é uma α-aproximação para o problema* 2-RCC.

Prova Não é difícil ver que a recoloração $(P, \widetilde{C})$ devolvida pelo algoritmo Aprox2RCC é uma recoloração convexa de (P, C), já que esta é quase igual àquela devolvida pelo algoritmo AproxRCC-Forte aplicado a (P', C'). Como os vértices que estão em P, mas não estão em P' (os coloridos fracamente), recebem a cor do par forte correspondente, isso garante a convexidade da coloração $\widetilde{C}$.

Vamos agora provar que o algoritmo Aprox2RCC preserva a razão de aproximação α do algoritmo AproxRCC-Forte.

É imediato que o custo da recoloração $(P, \widetilde{C})$ devolvida pelo algoritmo Aprox2RCC é igual ao custo da recoloração $(P', \widehat{C})$ (devolvida pelo algoritmo AproxRCC-Forte) acrescido de k. De fato, essas k trocas adicionais devem ser computadas porque os vértices coloridos fracamente tiveram sua cor trocada (veja o passo 5 do Algoritmo Aprox2RCC). Temos então as seguintes desigualdades:

$$\begin{aligned}
\text{custo}(\widetilde{C}) &= \text{custo}(\widehat{C}) + k \\
&\leq \alpha \ \text{opt}(P', C') + k && (4.1)\\
&= \alpha \left(\text{opt}(P, C) - k\right) + k && (4.2)\\
&= \alpha \ \text{opt}(P, C) - (\alpha - 1)k \\
&\leq \alpha \ \text{opt}(P, C). && (4.3)
\end{aligned}$$

A primeira desigualdade 4.1 acima segue do fato de que $(P', \widehat{C})$ foi obtido pelo algoritmo AproxRCC-Forte aplicado a (P', C'), que é uma α-aproximação. A igualdade 4.2 segue do Lema 4.1.2, e a desigualdade 4.3 segue do fato de que $\alpha \geq 1$. Concluímos assim que o algoritmo Aprox2RCC é uma α-aproximação para o 2-RCC. Vale notar que se o algoritmo AproxRCC-Forte é polinomial, então Aprox2RCC é polinomial. □

Juntando os resultados vistos anteriormente, temos o seguinte resultado para o algoritmo Aprox2RCC.

Teorema 4.2.4 *O algoritmo* Aprox2RCC *é uma* 3/2*-aproximação para o problema* 2-RCC*. Além disso, a razão* 3/2 *é justa.*

Prova A razão de aproximação segue do Teorema 4.2.1 e dos lemas 4.1.1 e 4.2.3. O exemplo dado (Figura 4.5)para mostrar que a razão 3/2 é justa para o algoritmo AproxRCC-Forte, mostra que razão 3/2 também é justa para Aprox2RCC. □

4.3 Sobre o algoritmo de aproximação para RCC

Como já mencionamos, Moran e Snir [17], projetaram um algoritmo de 2-aproximação para o problema RCC, de complexidade $O(|\mathcal{C}|n)$. Chamamos este algoritmo de AproxRCC-MS.

O algoritmo que projetamos para o problema 2-RCC foi inspirado no algoritmo de Moran e Snir que se baseia na ideia de multas e blocos, conforme apresentamos na descrição do algoritmo AproxRCC-Forte. A diferença básica é na definição de multa de uma cor, que Moran e Snir definiram como segue.

Se Q é um subcaminho de P, definimos a **multa de** d **sobre** Q**, com respeito a** C, como

$$\text{multa}_{d,C}(Q) = |V(Q) \cap \overline{C[d]}| + |\overline{V(Q)} \cap C[d]|.$$

Ou seja, $\text{multa}_{d,C}(Q)$ é calculada somando-se o número de vértices em Q que não possuem a cor d e o número de vértices de cor d que não estão em Q.

Os conceitos de **multa mínima**, **bloco**, **vértice livre** são definidos como apresentamos para o algoritmo Aprox-RCC-Forte. No caso do algoritmo Aprox-RCC-MS, quando há mais de um subcaminho de multa mínima para uma cor d, pode-se tomar qualquer um deles para ser o bloco B_d.

Vale aqui destacar que Moran e Snir consideraram a versão com pesos do problema RCC, que chamamos de RCC-Pesos. Assim, vamos considerar agora o algoritmo de 2-aproximação para o RCC-Pesos, denotado a seguir por AproxRCC-MS-Pesos.

Lembramos que, no problema RCC-Pesos, a instância consiste em uma tripla (P, C, w), onde w é uma função peso que atribui para cada vértice v um valor real não-negativo $w(v)$, definido como o **peso** de v. Para um conjunto de vértices X, o **peso de** X, denotado por $w(X)$, é dado por $w(X) = \sum_{v \in X} w(v)$. Neste caso, o **custo de uma recoloração convexa** C' de C é $custo_C(C') = w(X(C'))$, onde $X(C') = \{v \in P : C(v) \neq C'(v)\}$. Dizemos que C' é uma **recoloração convexa ótima** se C' é uma recoloração convexa de custo mínimo.

Seja (P, C, w) um caminho colorido com pesos. Para uma cor d e um subcaminho Q de P, definimos a **multa de** d **sobre** Q**, com respeito a** C, como sendo

$$multa_{d,C}(Q) = w(V(Q) \cap \overline{C[d]}) + w(\overline{V(Q)} \cap C[d]).$$

Observe que, $\text{multa}_{d,C}(Q)$ é a soma de duas parcelas: uma delas é a soma dos pesos dos vértices que pertencem ao subcaminho Q e não estão coloridos com a cor d, e a outra parcela é a soma dos pesos dos vértices que não pertencem ao subcaminho Q e estão coloridos com a cor d. Tal como convencionamos antes, quando a coloração está clara pelo contexto, escrevemos simplesmente $\text{multa}_d(Q)$.

Note que, quando $w(v) = 1$ para todo $v \in P$, temos exatamente a definição de multa para o AproxRCC-MS que mencionamos no início desta seção.

O algoritmo AproxRCC-MS-Pesos usa uma subrotina (linear), apenas citada em [17], que descrevemos a seguir, chamada aqui de AchaBloco-Pesos. Essa subrotina encontra, para um dado caminho colorido (P, C, w) e uma cor d, um bloco B_d, ou seja, um sub-

caminho de P em relação ao qual a $multa_d$ é mínima (tem valor m_d^*). Tal subrotina está fundamentada no seguinte fato, cuja prova é bem simples:

O problema de encontrar um bloco B_d é equivalente ao problema de encontrar um subcaminho Q para o qual a diferença entre a soma dos pesos dos vértices de cor d e a soma dos pesos dos vértices de cor diferente de d é máxima. De fato, basta notar que vale a seguinte igualdade.

$$\begin{aligned}\text{multa}_{d,C}(Q) &= w\Big(V(Q) \cap \overline{C[d]}\Big) + w\Big(\overline{V(Q)} \cap C[d]\Big)\\ &= w\Big(V(Q) \cap \overline{C[d]}\Big) + \Big(w\Big(V(P) \cap C[d]\Big) - w\Big(V(Q) \cap C[d]\Big)\Big)\\ &= w\Big(V(P) \cap C[d]\Big) - \Big(w\Big(V(Q) \cap C[d]\Big) - w\Big(V(Q) \cap \overline{C[d]}\Big)\Big).\end{aligned}$$

Como $w(V(P) \cap C[d])$ tem um valor fixo, encontrar Q tal que a $\text{multa}_{d,C}(Q)$ é mínima é equivalente a encontrar Q para o qual a parcela $w\Big(V(Q) \cap C[d]\Big) - w\Big(V(Q) \cap \overline{C[d]}\Big)$ é máxima. É exatamente esse último problema que é resolvido na subrotina que segue.

Algoritmo **AchaBloco-Pesos**

Entrada: Uma quádrupla (P, C, w, d), onde $P = (v_1, \ldots, v_n)$ e d é uma cor.

Saída: Um bloco B_d em relação ao qual a multa de d é mínima.

```
SomaMax ← 0;  SomaAtual ← 0
j ← 1;  k ← 1;  p ← 1; /* k = início e p = fim do bloco procurado */
enquanto j ≤ n faça
    se C(v_i) = d
        então  SomaAtual ← SomaAtual + w(v_i)
        senão  SomaAtual ← SomaAtual - w(v_i)
    se SomaAtual > SomaMax
        então   SomaMax ← SomaAtual;  p ← j
        senão se SomaAtual < 0
            então SomaAtual ← 0;  k ← j + 1;  p ← j + 1
    j ← j + 1
devolva B_d = (v_k, ..., v_p)
```

Note que, tal como no caso da seção anterior, para cada cor d, o bloco B_d encontrado pelo algoritmo acima sempre começa num vértice com a cor d. Porém, neste caso, o primeiro vértice de P não necessariamente pertence a um bloco: pode ser um vértice livre.

Em linhas gerais, o algoritmo AproxRCC-MS-Pesos percorre o caminho P a partir do seu início, e faz uso dos blocos para decidir a cor a ser atribuída aos vértices. Como os vértices iniciais podem não pertencer a algum bloco, esses vértices recebem a cor do primeiro bloco que é encontrado (veja o passo 2). O restante é análogo ao que já descrevemos para o algoritmo AproxRCC-Forte.

Algoritmo **AproxRCC-MS-Pesos**

Entrada: Uma tripla (P, C, w), onde $P = (v_1, \ldots, v_n)$.

Saída: Uma coloração convexa $\widehat{C}$ de P.

01 Para cada cor $d \in \mathcal{C}$, seja $B_d \leftarrow$ AchaBloco-Pesos (P, C, w, d)

02 Seja v_t o primeiro vértice em P que não é livre; seja $d \leftarrow C(v_t)$

03 **para** $i = 1, \ldots, n$ **faça**

04 **se** $v_i \in B_d$ ou v_i é livre

05 **então** $\widehat{C}(v_i) \leftarrow d$;

06 **senão** // v_i `pertence a um bloco distinto de` B_d

07 seja k tal que $v_i \in B_k$

08 $\widehat{C}(v_i) \leftarrow k$; $d \leftarrow k$

09 **devolva** $\widehat{C}$

4.3.1 Análise de desempenho do algoritmo AproxRCC-MS-Pesos

Teorema 4.3.2 (Moran e Snir, 2005) *O algoritmo AproxRCC-MS-Pesos é uma* 2*-aproximação para o RCC-Pesos.*

Prova É fácil ver que a coloração $\widehat{C}$ é convexa. Afirmamos que a coloração $\widehat{C}$ é tal que custo$(\widehat{C}) \leq \sum_d m_d^*$. De fato,

$$\text{custo}(\widehat{C}) = \sum \{w(v_i) : C(v_i) \neq \widehat{C}(v_i)\},$$

e portanto, $w(v_i)$ é computado em *custo*$(\widehat{C})$ somente se $C(v_i) = d$ e $\widehat{C}(v_i) = d'$ para $d \neq d'$. Pelo algoritmo, $\widehat{C}(v_i) = d'$ somente se $v_i \in B_{d'}$ ou v_i é livre. No primeiro caso, $w(v_i)$

é contabilizado em $m^*_{d'}$. No segundo caso, $w(v_i)$ é contabilizado em m^*_d. Em ambos os casos, $w(v_i)$ ocorre na soma $\sum_{d\in\mathcal{C}} m^*_d$. Logo,

$$\text{custo}(\widehat{C}) \leq \sum_d m^*_d.$$

Seja $(P, \widetilde{C})$ uma recoloração convexa qualquer de (P, C). Para cada cor d, denote por $\widetilde{Q}_d$ o subcaminho de P formado pelos vértices de cor d em $\widetilde{C}$. Então temos que

$$\begin{aligned}
\sum_{d\in\mathcal{C}} m^*_d &\leq \sum_{d\in\mathcal{C}} \text{multa}_d(\widetilde{Q}_d) \\
&= \sum_{d\in\mathcal{C}} w\Big(\{v \in V(P) : \widetilde{C}(v) = d \text{ e } C(v) \neq d\} \cup \{v \in V(P) : \widetilde{C}(v) \neq d \text{ e } C(v) = d\}\Big) \\
&= 2\, w\Big(\{v \in V(P) : \widetilde{C}(v) \neq C(v)\}\Big) \\
&= 2\,\text{custo}(\widetilde{C}).
\end{aligned}$$

Logo, para qualquer recoloração convexa $\widetilde{C}$ de C, temos que

$$\text{custo}(\widehat{C}) \leq \textstyle\sum_d m^*_d \leq 2\,\text{custo}(\widetilde{C}).$$

Esta conclusão vale em particular para uma recoloração convexa ótima. Como o algoritmo ApoxRCC-SM-Pesos é polinomial (quadrático no comprimento do caminho dado), segue que ele é uma 2-aproximação (polinomial). □

$$\begin{aligned}
(P, C) :&\ \overbrace{a\,a\,a\,a}^{B_a}\ \underbrace{b\,b\ldots b\,b}_{k-1}\ \underbrace{c\,c\ldots c\,c}_{k}\ \overbrace{\underbrace{b\,b\ldots b\,b}_{k+1}}^{B_b}\ \overbrace{\underbrace{c\,c\ldots c\,c}_{k+2}}^{B_c} \\
(P, \widehat{C}) :&\ a\,a\,a\,a\ \underbrace{a\,a\ldots a\,a}_{k-1}\ \underbrace{a\,a\ldots a\,a}_{k}\ \underbrace{b\,b\ldots b\,b}_{k+1}\ \underbrace{c\,c\ldots c\,c}_{k+2} \\
(P, C^*) :&\ a\,a\,a\,a\ \underbrace{b\,b\ldots b\,b}_{k-1}\ \underbrace{b\,b\ldots b\,b}_{k}\ \underbrace{b\,b\ldots b\,b}_{k+1}\ \underbrace{c\,c\ldots c\,c}_{k+2}
\end{aligned}$$

Figura 4.6: *A razão 2 do algoritmo AproxRCC-MS é justa.*

Proposição 4.3.3 *A razão de aproximação* 2 *do algoritmo AproxRCC-MS (e portanto do algoritmo AproxRCC-MS-Pesos) é justa.*

Prova Considere a instância (P, C) do problema RCC mostrada na Figura 4.6.

Note que, o algoritmo AproxRCC-MS devolve uma recoloração $\widehat{C}$ que faz $2k - 1$ trocas de cor, enquanto uma recoloração convexa ótima C^* faz k trocas. Neste caso, a razão é $2k - 1/k$, e portanto tende a 2 quando k cresce. □

Observamos que Moran e Snir [17] nada mencionaram a respeito de a razão 2 ser justa. Também nada mencionaram a respeito de resultados computacionais obtidos com este algoritmo. Testamos a versão sem pesos, conforme mostraremos no Capítulo 6. Os resultados experimentais para instâncias geradas aleatoriamente mostraram que na prática o desempenho desse algoritmo é muito bom: a razão de aproximação que obtivemos é próxima de $1,2$.

Capítulo 5

Estudo do poliedro associado ao problema RCC

Neste capítulo estudamos o problema da Recoloração Convexa de Caminhos, RCC, sob o ponto de vista poliédrico. Associamos um poliedro ao RCC, e estudamos sua estrutura facial, com vistas ao desenvolvimento de um algoritmo baseado nesses resultados. Como resultados dessa natureza não foram encontrados na literatura, achamos que tais estudos seriam interessantes, tanto do ponto de vista teórico quanto prático.

Modelar o RCC como um problema de programação linear inteira não é difícil. Porém, uma tal formulação, pode não ser muito útil do ponto de vista prático, já que, em geral, resolver um PL/01 é um problema NP-difícil. Veremos no próximo capítulo como podemos utilizar a relaxação de uma tal formulação para desenvolver uma heurística para o RCC. Neste capítulo, o foco central é o estudo do poliedro $\mathcal{P}_{nk} = \mathcal{P}_{nk}(P, C) \subseteq \mathbb{R}^{nk}$, que definimos como o fecho convexo dos vetores de incidência de recolorações convexas de uma dada instância (P, C).

Exibimos aqui várias facetas desse poliedro, que se mostraram bastante úteis no algoritmo *branch-and-cut* que desenvolvemos para o RCC.

5.1 Uma formulação linear inteira do problema RCC

Para formular o RCC como um programa linear inteiro, consideramos que uma instância desse problema consiste em um par (P, C), onde P é um caminho e C é uma coloração de P. Denotamos por $V = \{1, 2, \ldots, n\}$ o conjunto de vértices de P e por $\mathcal{C}$ o conjunto de cores de C. Convencionamos, neste capítulo, que $P = (1, 2, \ldots, n)$ e $\mathcal{C} = \{1, 2, \ldots, k\}$. Além disso, supomos que $n \geq 3$ e $k \geq 3$; caso contrário a correspondente instância pode ser facilmente resolvida.

Para cada vértice $i \in V$ e cor $c \in \mathcal{C}$, consideramos uma variável binária x_{ic} com o

seguinte significado:

$$x_{ic} = \begin{cases} 1 & \text{se o vértice } i \text{ recebe a cor } c, \\ 0 & \text{caso contrário.} \end{cases}$$

Para expressar a função objetivo, consideramos constantes w_{ic} para $i \in V$ e $c \in \mathcal{C}$, definidas como segue:

$$w_{ic} = \begin{cases} 1 & \text{se o vértice } i \text{ tem a cor } c, \\ 0 & \text{caso contrário.} \end{cases}$$

Neste caso, obtemos uma formulação 0/1 para o RCC como um problema de maximização como segue:

$$\text{maximize} \sum_{i \in V} \sum_{c \in \mathcal{C}} w_{ic}\, x_{ic}$$

$$\text{sujeito a} \quad \sum_{c \in \mathcal{C}} x_{ic} = 1 \qquad i \in V \tag{5.1}$$

$$x_{pc} - x_{qc} + x_{rc} \leq 1 \qquad 1 \leq p < q < r \leq n,\ c \in \mathcal{C} \tag{5.2}$$

$$x_{ic} \in \{0, 1\} \qquad i \in V,\ c \in \mathcal{C}. \tag{5.3}$$

Observamos que, nessa formulação, o objetivo é maximizar o número de vértices de P que têm suas cores mantidas (que é equivalente a minimizar o número de vértices que têm suas cores trocadas). Ou seja, neste caso, $\text{opt}(P, C) = n -$ valor ótimo do PL/01.

As equações (5.1) impõem que cada vértice deve receber exatamente uma das cores em $\mathcal{C}$. O conjunto de inequações (5.2) impõe que a coloração seja convexa. Mais especificamente, para cada cor $c \in \mathcal{C}$, se os vértices p e r, onde $r \geq p + 2$, recebem a cor c, então os vértices q, para $q = p + 1, \ldots, r - 1$, também devem receber a cor c.

5.2 O poliedro associado ao problema RCC

Vamos agora associar um poliedro ao RCC e estudar sua estrutura facial. Para isso, precisamos estabelecer a notação que será usada a seguir.

Seja (P, C') uma recoloração convexa de (P, C), então denotamos por $\chi(C') \in \{0, 1\}^{nk}$ o **vetor de incidência de** C'. Ou seja,

$$\chi(C') = (x_{ic} : 1 \leq i \leq n, 1 \leq c \leq k) = \boxed{x_{11}, x_{12}, \ldots, x_{1k},} \boxed{x_{21}, \ldots, x_{2k},} \boxed{\ldots} \boxed{x_{n1}, \ldots, x_{nk}}.$$

Por exemplo, se $P = (1,2,3,4,5,6,7)$, $\mathcal{C} = \{1,2,3,4\}$ e C' é a coloração convexa

$$C' = \boxed{2}\boxed{2}\boxed{4}\boxed{4}\boxed{3}\boxed{3}\boxed{1},$$

então

$$\chi(C') = \boxed{0,1,0,0}\boxed{0,1,0,0}\boxed{0,0,0,1,}\boxed{0,0,0,1,}\boxed{0,0,1,0}\boxed{0,0,1,0}\boxed{1,0,0,0} \in \mathbb{R}^{28}.$$

Seja $\mathcal{P}_{nk} := \mathcal{P}_{nk}(P,C)$ o **poliedro associado ao problema** RCC definido como

$$\mathcal{P}_{nk} = \text{conv}\{x \in \mathbb{R}^{nk} : x \text{ é vetor de incidência de uma recoloração convexa de } (P,C)\}.$$

O próximo resultado segue facilmente da formulação linear 0/1 que exibimos anteriormente.

Proposição 5.2.1 $\mathcal{P}_{nk} = \text{conv}\{x \in \mathbb{R}^{nk} : x \text{ satisfaz } (5.1), (5.2) \text{ e } (5.3)\}$.

Prova Seja C' uma recoloração convexa de (P,C) e $\chi(C') = (x_{ic} : 1 \leq i \leq n, 1 \leq c \leq k)$ o vetor de incidência de C'. Vamos mostrar que $\chi(C')$ satisfaz (5.1), (5.2) e (5.3). Claramente, $\chi(C')$ satisfaz (5.1) e (5.3). Considere uma cor $c \in \mathcal{C}$. Se dois vértices p e r em P, onde $r \geq p+2$, são tais que $C'(p) = C'(r) = c$, então como C' é uma coloração convexa, segue que todo vértice q, onde $p < q < r$, deve ter a cor c. Ou seja, se $x_{pc} = 1$ e $x_{rc} = 1$, então $x_{qc} = 1$ para todo $p < q < r$. Portanto, $\chi(C')$ satisfaz a inequação (5.2).

Reciprocamente, se x é um vetor em $\mathbb{R}^{nk}$ que satisfaz (5.1), (5.2) e (5.3), então claramente x é vetor de incidência de uma recoloração convexa de (P,C). De fato, as inequações satisfeitas por x indicam que cada vértice em P recebe exatamente uma cor, e que a coloração de P definida por x é convexa, já que a inequação (5.2) está satisfeita.

Com isso, concluímos que $\mathcal{P}_{nk}$ é o fecho convexo dos vetores em $\mathbb{R}^{nk}$ que satisfazem (5.1), (5.2) e (5.3).

□

Nas próximas seções apresentaremos alguns resultados a respeito do poliedro $\mathcal{P}_{nk}$.

Vamos denotar por $Mx = \mathbb{1}$ o sistema de equações (5.1) presente na formulação 0/1 que apresentamos para o problema RCC. Mais precisamente, M é uma matriz em $\{0,1\}^{n \times nk}$ da forma exibida a seguir, onde $\mathbb{1}_k = \underbrace{1\,1\ldots1}_{k}$ e $0_k = \underbrace{0\,0\ldots0}_{k}$.

$$M = \begin{bmatrix} \mathbb{1}_k & 0_k & 0_k & \dots & 0_k \\ 0_k & 1_k & 0_k & \dots & 0_k \\ 0_k & 0_k & 1_k & \dots & 0_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_k & 0_k & 0_k & \dots & 1_k \end{bmatrix}$$

Esta representação facilitará o entendimento das provas dos próximos resultados.

Primeiramente notamos que $Mx = \mathbb{1}$ define afim($\mathcal{P}_{nk}$), conforme mostramos em seguida. Antes de apresentar a prova, lembramos que para um conjunto S, define-se afim(S) como sendo o conjunto formado por todas as combinações afins de elementos de S.

Teorema 5.2.2 *Temos que* afim($\mathcal{P}_{nk}$) $= \{x \in \mathcal{P}_{nk} : Mx = \mathbb{1}\}$.

Prova Vamos provar que as únicas restrições que são satisfeitas com igualdade pelos pontos de $\mathcal{P}_{nk}$ são combinações lineares de $Mx = \mathbb{1}$.

Seja $a^T x = a_0$ uma equação arbitrária que é satisfeita por qualquer ponto $x \in \mathcal{P}_{nk}$. Vamos mostrar que existe $\lambda = (\lambda_i) \in \mathbb{R}^n$ tal que $a^T = \lambda^T M$, implicando que, $a^T x = a_0$ é uma combinação linear das linhas de M.

Passo 1 - Considere as seguintes k colorações convexas C_j^1, para $j = 1, \dots, k$ (cores), e seus respectivos vetores de incidência $x_j^1 = \chi(C_j^1)$.

$C_1^1 =$	1	2	2	...	2	$x_1^1 =$	100...00	010...00	010...00	...	010...00
$C_2^1 =$	2	2	2	...	2	$x_2^1 =$	010...00	010...00	010...00	...	010...00
$C_3^1 =$	3	2	2	...	2	$x_3^1 =$	001...00	010...00	010...00	...	010...00
	⋮						⋮				
$C_k^1 =$	k	2	2	...	2	$x_k^1 =$	000...01	010...00	010...00	...	010...00

Mais formalmente, para $j = 1, \dots, k$, temos que

$$C_j^1(v) = \begin{cases} j & \text{se} \quad v = 1, \\ 2 & \text{se} \quad 2 \leq v \leq n. \end{cases}$$

Para $j = 1, \ldots, k$, temos que C_j^1 é uma coloração convexa, e portanto $x_j^1 \in \mathcal{P}_{nk}$. Logo, $a^T x_j^1 = a_0$, para $j = 1, \ldots, k$, ou seja,

$$\begin{aligned} a_{11} + a_{22} + & \ \ldots \ + a_{n2} = a_0 \\ a_{12} + a_{22} + & \ \ldots \ + a_{n2} = a_0 \\ & \ldots \\ a_{1k} + a_{22} + & \ \ldots \ + a_{n2} = a_0. \end{aligned}$$

Das equações acima, segue que $a_{11} = a_{12} = \ldots = a_{1k}$. Seja então

$$\beta_1 := a_{11} = a_{12} = \ldots = a_{1k}.$$

Passo 2 - Agora considere as seguintes colorações convexas C_j^2, para $j = 1, \ldots, k$, e seus respectivos vetores de incidência $x_j^2 = \chi(C_j^2)$.

$C_1^2 =$ | 1 | 1 | 2 | ... | 2 | $\quad x_1^2 =$ | 100...00 | 100...00 | 010...00 | ... | 010...00 |

$C_2^2 =$ | 1 | 2 | 2 | ... | 2 | $\quad x_2^2 =$ | 100...00 | 010...00 | 010...00 | ... | 010...00 |

$C_3^2 =$ | 1 | 3 | 2 | ... | 2 | $\quad x_3^2 =$ | 100...00 | 001...00 | 010...00 | ... | 010...00 |

$\vdots \qquad\qquad \vdots$

$C_k^2 =$ | 1 | k | 2 | ... | 2 | $\quad x_k^2 =$ | 100...00 | 000...01 | 010...00 | ... | 010...00 |

Mais formalmente, para $j = 1, \ldots, k$, temos que

$$C_j^2(v) = \begin{cases} 1 & \text{se} \quad v = 1, \\ j & \text{se} \quad v = 2, \\ 2 & \text{se} \quad 3 \leq v \leq n. \end{cases}$$

Novamente, como C_j^2 é uma coloração convexa para $j = 1, \ldots, k$, segue que $x_j^2 \in \mathcal{P}_{nk}$. Então $a^T x_j^2 = a_0$, para $j = 1, \ldots, k$, ou seja,

$$a_0 = a_{11} + a_{21} + \ldots + a_{n2} = a_{11} + a_{22} + \ldots + a_{n2} = \ldots = a_{11} + a_{2k} + \ldots + a_{n2}.$$

Portanto, $a_{21} = a_{22} = \ldots = a_{2k}$. Seja então

$$\beta_2 := a_{2j} = a_{22} = \ldots = a_{2k}.$$

Generalizando a ideia apresentada no Passo 1 e no Passo 2, podemos expressar de um modo unificado o Passo i da seguinte forma:

Passo i (onde $i \in \{1, \ldots, n\}$). Considere as seguintes colorações convexas C_j^i, para $j = 1, \ldots, k$, e seus respectivos vetores de incidência $x_j^i = \chi(C_j^i)$.

$$\begin{array}{rcccccccc} & & & & i & & & & \\ C_1^i = & \boxed{1} & \boxed{1} & \boxed{\ldots} & \boxed{1} & \boxed{2} & \boxed{\ldots} & \boxed{2} \\ C_2^i = & \boxed{1} & \boxed{1} & \boxed{\ldots} & \boxed{2} & \boxed{2} & \boxed{\ldots} & \boxed{2} \\ C_3^i = & \boxed{1} & \boxed{1} & \boxed{\ldots} & \boxed{3} & \boxed{2} & \boxed{\ldots} & \boxed{2} \\ & & & & \vdots & & & \\ C_k^i = & \boxed{1} & \boxed{1} & \boxed{\ldots} & \boxed{k} & \boxed{2} & \boxed{\ldots} & \boxed{2} \end{array}$$

No i-ésimo passo tomamos k colorações, onde a única diferença entre elas é a cor do vértice i, que varia de 1 a k. Além disso, note que os vértices que antecedem i têm sempre a cor 1, e os vértices que sucedem i têm sempre a cor 2.

Mais formalmente, para $j = 1, \ldots, k$, temos que

$$C_j^i(v) = \begin{cases} 1 & \text{se} \quad 1 \leq v < i, \\ j & \text{se} \quad v = i, \\ 2 & \text{se} \quad i < v \leq n. \end{cases}$$

Usando os argumentos vistos no Passo 1 e no Passo 2, concluímos que os coeficientes de a^T correspondentes a um vértice i são sempre iguais, ou seja,

$$\beta_i := a_{i1} = a_{i2} = \ldots = a_{ik} \quad \text{para} \quad i = 1, \ldots, n.$$

Assim,

$$a^T = (\underbrace{\beta_1, \ldots, \beta_1}_{k}, \underbrace{\beta_2, \ldots, \beta_2}_{k}, \ldots, \underbrace{\beta_n, \ldots, \beta_n}_{k}).$$

Portanto, existe $\lambda \in \mathbb{R}^n$ tal que $a^T = \lambda^T M$. Basta tomar $\lambda_i = \beta_i$ para $i = 1, \ldots, n$. □

Pelo Teorema 5.2.2 e Proposição 2.4.1, segue imediatamente o seguinte resultado.

Teorema 5.2.3 *A dimensão do poliedro $\mathcal{P}_{nk}$ é dada por* $\dim(\mathcal{P}_{nk}) = (k-1)n$.

Prova Pelo Teorema 5.2.2, sabemos que $\text{afim}(\mathcal{P}) = \{x \in \mathcal{P}_{nk} : Mx = \mathbb{1}\}$. É fácil ver que $\text{posto}(M) = n$, pois M possui n colunas que são linearmente independentes. Pela Proposição 2.4.1, $\dim(\mathcal{P}_{nk}) = nk - \text{posto}(M)$. Logo,

$$\dim(\mathcal{P}_{nk}) = nk - n = (k-1)n.$$

□

5.3 Inequações válidas para o poliedro $\mathcal{P}_{nk}$

Apresentamos nesta seção algumas inequações válidas para o poliedro $\mathcal{P}_{nk}$. Veremos, na próxima seção, que algumas delas definem facetas desse poliedro.

É claro que as inequações (5.1), (5.2) e $(0 \leq x_{ic} \leq 1)$ que definem uma relaxação linear de $\mathcal{P}_{nk}$ são válidas para $\mathcal{P}_{nk}$. O nosso interesse é encontrar outras inequações válidas, preferencialmente aquelas que definem facetas.

Uma das maneiras de se obter inequações válidas é fazer combinação linear de inequações válidas de modo a obter uma nova que seja gerada depois de um arredondamento. Essa técnica será usada a seguir.

Teorema 5.3.1 *[Inequações de convexidade-generalizada do tipo t]*
Seja $\varphi = (p, r, t)$, onde p, r, t são inteiros tais que $1 \leq p < r - 1 < n$ e $t \leq \lfloor \frac{r-p}{2} \rfloor$. Sejam p_i e q_i para $i = 1, \ldots, t$ inteiros tais que

$$p = p_1 < q_1 < p_2 < q_2 < \ldots < p_t < q_t < r.$$

Então para todo $c \in \mathcal{C}$, a inequação

$$\sum_{i=1}^{t} x_{p_i c} - \sum_{i=1}^{t} x_{q_i c} + x_{rc} \leq 1 \tag{5.4}$$

é válida para $\mathcal{P}_{nk}$.

Antes de provarmos este teorema, daremos mais detalhes sobre esta família de inequações. Primeiramente, note que, cada tripla $\varphi = (p, r, t)$ pode induzir uma ou mais inequações: para cada escolha de inteiros p_i e q_i, tais que $p = p_1 < q_1 < p_2 < q_2 < \ldots < p_t < q_t < r$, temos uma inequação correspondente. Nestas inequações as

posições p_i têm coeficiente 1 e as posições q_i têm coeficientes -1. Dizemos que uma tal inequação é **induzida por** $\varphi = (p, r, t)$. Também nos referiremos a essas inequações como **inequações de convexidade-generalizada do tipo** t, ou simplesmente **inequações de convexidade-generalizada**.

Exemplo 1: Considere $\varphi = (p, r, 1)$, onde $r \geq p + 2$.

Para $p = p_1 < q_1 < r$, a inequação correspondente, para $c \in \mathcal{C}$, é

$$x_{pc} - x_{q_1c} + x_{rc} \leq 1.$$

Considerando todas as possíveis escolhas de q_1, ou seja $p < q = q_1 < r$, para quaisquer p e r, temos as inequações (5.2), que já vimos anteriormente.

Exemplo 2: Considere $\varphi = (p, r, 2)$, onde $r \geq p + 4$.

Para cada escolha de q_1, p_2 e q_2 tais que $p = p_1 < q_1 < p_2 < q_2 < r$, temos uma inequação da forma

$$x_{pc} - x_{q_1c} + x_{p_2c} - x_{q_2c} + x_{rc} \leq 1.$$

Daremos a seguir um exemplo mais específico. Para isto, tome $n = 7$ e $\varphi = (1, 6, 2)$. Para este caso, temos as seguintes inequações, para cada cor $c \in \mathcal{C}$:

$$\begin{array}{cccccccc}
x_{1c} & -x_{2c} & +x_{3c} & -x_{4c} & & +x_{6c} & \leq & 1 \\
x_{1c} & -x_{2c} & +x_{3c} & & -x_{5c} & +x_{6c} & \leq & 1 \\
x_{1c} & -x_{2c} & & +x_{4c} & -x_{5c} & +x_{6c} & \leq & 1 \\
x_{1c} & & -x_{3c} & +x_{4c} & -x_{5c} & +x_{6c} & \leq & 1\,.
\end{array}$$

Prova do Teorema 5.3.1

Provaremos que as inequações de convexidade-generalizada do tipo t, $t \geq 2$, são válidas, usando o fato de que inequações de convexidade-generalizada do tipo j, onde $j < t$ são válidas.

As inequações de convexidade-generalizada do tipo 1, abaixo reproduzidas, são precisamente as inequações (5.2), que sabemos serem válidas para $\mathcal{P}_{nk}$, pela Proposição 5.2.1.

$$x_{pc} - x_{qc} + x_{rc} \leq 1 \qquad 1 \leq p < q < r \leq n, \;\; c \in \mathcal{C}.$$

Usando o fato de que inequações induzidas por $\varphi(p, r, 1)$ são válidas, vamos provar a validade de uma inequação induzida por $\varphi(p, r, 2)$. Para isto, fixe uma cor c, e considere a soma das seguintes três inequações válidas, onde $1 \leq p = p_1 < q_1 < p_2 < q_2 < r$.

$$\begin{array}{cccccccc}
x_{pc} & -x_{q_1c} & +x_{p_2c} & & & \leq & 1 \\
 & & +x_{p_2c} & -x_{q_2c} & +x_{rc} & \leq & 1 \\
x_{pc} & & & -x_{q_2c} & +x_{rc} & \leq & 1 \\
\hline
2x_{pc} & -x_{q_1c} & +2x_{p_2c} & -2x_{q_2c} & +2x_{rc} & \leq & 3\,.
\end{array}$$

Temos assim que a inequação $2x_{pc} - x_{q_1c} + 2x_{p_2c} - 2x_{q_2c} + 2x_{rc} \leq 3$ é válida para $\mathcal{P}_{nk}$. Somando essa inequação com a inequação de não-negatividade $-x_{q_1c} \leq 0$, obtemos

$$2x_{pc} - 2x_{q_1c} + 2x_{p_2c} - 2x_{q_2c} + 2x_{rc} \leq 3.$$

Dividindo a inequação acima por 2, obtemos:

$$x_{pc} - x_{q_1c} + x_{p_2c} - x_{q_2c} + x_{rc} \leq \frac{3}{2}.$$

Como os vértices do poliedro $\mathcal{P}_{nk}$ são vetores binários, podemos arredondar o termo à direita da inequação acima, e obter a seguinte inequação mais forte, que certamente é válida para $\mathcal{P}_{nk}$:

$$x_{pc} - x_{q_1c} + x_{p_2c} - x_{q_2c} + x_{rc} \leq 1. \tag{5.5}$$

Provaremos a seguir a validade de uma inequação induzida por $\varphi(p, r, 3)$, para ilustrar melhor o processo de combinação e arredondamento. Uma prova para um t qualquer será feita depois.

Para $t = 3$, fixe uma cor c, e considere as seguintes inequações válidas, onde $1 \leq p = p_1 < q_1 < p_2 < q_2 < p_3 < q_3 < r$.

Considere a inequação obtida pela soma das seguintes quatro inequações válidas para $\mathcal{P}_{nk}$:

$$\begin{array}{ccccccccc}
x_{pc} & -x_{q_1c} & +x_{p_2c} & -x_{q_2c} & +x_{p_3c} & & & \leq & 1 \\
 & & +x_{p_2c} & -x_{q_2c} & +x_{p_3c} & -x_{q_3c} & +x_{rc} & \leq & 1 \\
x_{pc} & & & & & -x_{q_3c} & +x_{rc} & \leq & 1 \\
 & -x_{q_1c} & & & & & & \leq & 0 \\
\hline
2x_{pc} & -2x_{q_1c} & +2x_{p_2c} & -2x_{q_2c} & +2x_{p_3c} & -2x_{q_3c} & +2x_{rc} & \leq & 3\,.
\end{array}$$

Dividindo a inequação acima por 2 e fazendo o arredondamento, obtemos:

$$x_{pc} - x_{q_1c} + x_{p_2c} - x_{q_2c} + x_{p_3c} - x_{q_3c} + x_{rc} \leq 1. \tag{5.6}$$

Procedendo de forma análoga ao que fizemos para os casos $t = 2$ e $t = 3$, podemos provar que inequações induzidas por $\varphi(p, r, t)$, $t > 4$, são válidas para $\mathcal{P}_{nk}$. Note que para obter a validade para o caso $\varphi(p, r, t)$, para a escolha de índices $1 \leq p < q_1 < p_2 <$

$q_2 < \ldots < p_{t-1} < q_{t-1} < p_t < q_t < r \leq n$, somamos quatro inequações, a saber:

- a inequação induzida por $\varphi(p, p_t, t-1)$, para a escolha de índices $p < q_1 < p_2 < q_2 < \ldots < p_{t-1} < q_{t-1} < p_t$;
- a inequação induzida por $\varphi(p_2, r, t-1)$, para a escolha de índices $p_2 < q_2 < \ldots < p_t < q_t < r$;
- a inequação $x_{pc} - x_{q_t c} + x_{rc} \leq 1$;
- e a inequação de não-negatividade $-x_{q_1 c} \leq 0$.

Depois disso, dividimos por 2 a inequação obtida, e fazemos o arredondamento.

□

Observamos que o número de inequações de convexidade-generalizada do tipo t cresce com n. Para cada t, estas são obtidas de um processo de combinação e arredondamento de inequações do tipo $t-1$, cada uma das quais definem facetas, como veremos na próxima seção. Podemos ter inequações desse tipo para $t = 1, 2, \ldots, \lfloor \frac{n-1}{2} \rfloor$. Esse fato é um forte indício de que o posto de Chvátal (*Chvátal rank*) de $\mathcal{P}_{nk}$ seja $O(n)$. Não explicaremos aqui este conceito, mas o leitor interessado poderá consultar o livro de Schrijver [22]. Vale ressaltar que Bockmayr, Eisenbrand, Hartmann e Schulz [4] provaram que o posto de Chvátal de qualquer politopo $\mathcal{P} \subseteq [0,1]^d$ é no máximo $O(d^3 \log d)$.

Exibimos a seguir soluções que satisfazem as inequações de convexidade-generalizada do tipo t e não satisfazem uma inequação de convexidade-generalizada do tipo $t+1$.

Exemplo para $t = 1$:

Seja x uma solução do PL relaxado, tal que

$$(x_{1c}, x_{2c}, x_{3c}, x_{4c}, x_{5c}) = (0.6, 0.3, 0.6, 0.3, 0.6).$$

Para a cor c, essa solução satisfaz as inequações do tipo 1, mas não satisfaz a inequação (do tipo 2) $x_{1c} - x_{2c} + x_{3c} - x_{4c} + x_{5c} \leq 1$.

Exemplo para $t = 2$:

Seja x uma solução do PL relaxado, tal que

$$(x_{1c}, x_{2c}, \ldots, x_{7c}) = (0.5, 0.3, 0.5, 0.3, 0.5, 0.3, 0.5).$$

Para a cor c, essa solução satisfaz as inequações do tipo 1 e do tipo 2, mas não satisfaz a inequação (do tipo 3) $x_{1c} - x_{2c} + x_{3c} - x_{4c} + x_{5c} - x_{6c} + x_{7c} \leq 1$.

5.4 Inequações que definem facetas do poliedro $\mathcal{P}_{nk}$

Nesta seção provamos quais das inequações válidas para o poliedro $\mathcal{P}_{nk}$ — vistas na seção anterior — induzem facetas. Analisamos as inequações triviais e também as inequações de convexidade-generalizada do tipo t.

O seguinte resultado básico (veja [11]) será usado para provar que certas inequações definem faceta de $\mathcal{P}_{nk}$.

Proposição 5.4.1 *Seja $\mathcal{P} = \mathcal{P}(A, b)$ um poliedro e F uma face não-trivial de $\mathcal{P}$. As seguintes afirmações são equivalentes:*

a) *F é uma faceta de $\mathcal{P}$;*

b) *$dim(F) = dim(\mathcal{P}) - 1$;*

c) *Seja $a^T x \leq a_0$ uma inequação válida para $\mathcal{P}$ tal que $F = \{x \in P : a^T x = a_0\}$. Então para toda inequação $b^T x \leq b_0$ válida para $\mathcal{P}$, tal que $F \subseteq \{x \in \mathcal{P} : b^T x = b_0\}$, temos que existe $\alpha \in \mathbb{R}$, $\alpha \geq 0$, e um vetor $\lambda \in \mathbb{R}^{ig(\mathcal{P})}$, tal que $b^T = \alpha a^T + \lambda^T A_{ig(\mathcal{P})}$ (e $b_0 = \alpha a_0 + \lambda^T b_{ig(\mathcal{P})}$), onde $A_{ig(\mathcal{P})}$ é conjunto das inequações de $Ax \leq b$ que são satisfeitas pelos pontos de $\mathcal{P}$ com igualdade.*

Como consequência do Teorema 5.2.2 e Proposição 2.4.1, podemos reescrever a proposição acima no caso específico do poliedro $\mathcal{P}_{nk}$ da seguinte forma.

Proposição 5.4.2 *Seja $\mathcal{P}_{nk}$ o poliedro para o problema RCC e F uma face não-trivial de $\mathcal{P}_{nk}$. As seguintes afirmações são equivalentes:*

a) *F é uma faceta de $\mathcal{P}_{nk}$;*

b) *$dim(F) = (k-1)n - 1$;*

c) *Seja $a^T x \leq a_0$ uma inequação válida para $\mathcal{P}_{nk}$ tal que $F = \{x \in \mathcal{P}_{nk} : a^T x = a_0\}$. Então para toda inequação $b^T x \leq b_0$ válida para $\mathcal{P}_{nk}$, tal que $F \subseteq \{x \in \mathcal{P}_{nk} : b^T x = b_0\}$, temos que existe $\alpha \in \mathbb{R}$, $\alpha \geq 0$, e um vetor $\lambda \in \mathbb{R}^n$, tal que $b^T = \alpha a^T + \lambda^T M$, onde M é a matriz definida na Seção 5.2.*

Primeiramente, vamos analisar as inequações de não-negatividade. Como provaremos a seguir, estas definem facetas do poliedro $\mathcal{P}_{nk}$.

Teorema 5.4.3 *Para todo vértice $i \in V$ e toda cor $c \in \mathcal{C}$, a inequação $x_{ic} \geq 0$ define uma faceta do poliedro $\mathcal{P}_{nk}$.*

Prova Fixe $i \in V$ e uma cor c. Seja $a^T x \geq a_0$ a inequação correspondente a $x_{ic} \geq 0$.

Seja $b^T x \geq b_0$ uma inequação válida para $\mathcal{P}_{nk}$ tal que

$$F_a := \{x \in \mathcal{P}_{nk} : a^T x = a_0\} \subseteq F_b := \{x \in \mathcal{P}_{nk} : b^T x = b_0\}.$$

Para mostrar que $a^T x \geq a_0$ define uma faceta de $\mathcal{P}_{nk}$, pela Proposição 5.4.2, precisamos provar que existe uma constante $\alpha \geq 0$, $\alpha \in \mathbb{R}$ e um vetor $\lambda \in \mathbb{R}^n$ tais que $b^T = \alpha a^T + \lambda^T M$.

Mostraremos agora a relação existente entre os coeficientes de b, onde

$$b^T = (b_{11}, \ldots, b_{1k}, b_{21}, \ldots, b_{2k}, \ldots, b_{i1}, \ldots, b_{ic}, \ldots, b_{ik}, \ldots, b_{n1}, \ldots, b_{nk}).$$

Suponha, sem perda de generalidade, que $c \neq \{1, 2\}$. (Podemos supor isso, pois $k \geq 3$. Em princípio, no lugar de 1 e 2 podemos tomar quaisquer duas cores distintas de c.)

Considere as seguintes colorações convexas e seus respectivos vetores de incidência (tal como fizemos na prova do Teorema 5.2.2).

$$C_1^1 = \boxed{1}\boxed{2}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_1^1 = \boxed{100\ldots00}\boxed{010\ldots00}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

$$C_2^1 = \boxed{2}\boxed{2}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_2^1 = \boxed{010\ldots00}\boxed{010\ldots00}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$C_c^1 = \boxed{c}\boxed{2}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_c^1 = \boxed{0\ldots010\ldots0}\boxed{010\ldots00}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$C_k^1 = \boxed{k}\boxed{2}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_k^1 = \boxed{000\ldots01}\boxed{010\ldots00}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

Cada um desses vetores de incidência pertence a F_a pois satisfaz a equação $x_{ic} = 0$.

Como $F_a \subseteq F_b$, cada um desses vetores satisfaz $b^T x = b_0$. Isso implica que

$$\begin{aligned}
b_{11} + b_{22} + \ \ldots \ + b_{n2} &= b_0 \\
b_{12} + b_{22} + \ \ldots \ + b_{n2} &= b_0 \\
&\vdots \\
b_{1c} + b_{22} + \ \ldots \ + b_{n2} &= b_0 \\
&\vdots \\
b_{1k} + b_{22} + \ \ldots \ + b_{n2} &= b_0.
\end{aligned}$$

Das equações acima, segue que $b_{11} = b_{12} = \ldots = b_{1c} = \ldots = b_{1k}$. Denote por β_1 esses coeficientes comuns (relativos ao vértice 1 e a todas as cores).

Agora tome as seguintes colorações convexas e seus vetores de incidência:

$$C_1^2 = \boxed{1}\boxed{1}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_1^2 = \boxed{100\ldots00}\boxed{100\ldots00}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

$$C_2^2 = \boxed{1}\boxed{2}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_2^2 = \boxed{100\ldots00}\boxed{010\ldots00}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$C_c^2 = \boxed{1}\boxed{c}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_c^2 = \boxed{100\ldots00}\boxed{0\ldots010\ldots0}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$C_k^2 = \boxed{1}\boxed{k}\boxed{2}\boxed{\ldots}\boxed{2} \qquad x_k^2 = \boxed{100\ldots00}\boxed{000\ldots01}\boxed{010\ldots00}\boxed{\ldots}\boxed{010\ldots00}$$

Cada um desses vetores de incidência pertence a F_a e portanto a F_b. Logo, eles satisfazem a equação $b^T x = b_0$, donde segue que

$$\begin{aligned}
b_{11} + b_{21} + \ \ldots \ + b_{n2} &= b_0 \\
b_{11} + b_{22} + \ \ldots \ + b_{n2} &= b_0 \\
&\vdots \\
b_{11} + b_{2c} + \ \ldots \ + b_{n2} &= b_0 \\
&\vdots \\
b_{11} + b_{2k} + \ \ldots \ + b_{n2} &= b_0.
\end{aligned}$$

Portanto, $b_{21} = b_{22} = \ldots = b_{2c} = \ldots = b_{2k}$. Analogamente, denote por β_2 esses coeficientes comuns (relativos ao vértice 2).

Procedendo de forma análoga, para todo j, $j \neq i$, concluímos que

$$b_{j1} = b_{j2} = \ldots = b_{jk} \quad \text{para todo} \quad j \neq i.$$

Para estudar b_{i*}, considere os vetores de incidência das seguintes colorações convexas.

$$\begin{array}{c} \\ C_1^i = \\ C_2^i = \\ \\ \\ C_{c-1}^i = \\ C_{c+1}^i = \\ \\ \\ C_k^i = \end{array} \begin{array}{c} i \\ \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & \ldots & 1 & 2 & \ldots & 2 \\ \hline 1 & 1 & \ldots & 2 & 2 & \ldots & 2 \\ \hline \end{array} \\ \vdots \\ \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & \ldots & c-1 & 2 & \ldots & 2 \\ \hline 1 & 1 & \ldots & c+1 & 2 & \ldots & 2 \\ \hline \end{array} \\ \vdots \\ \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 1 & \ldots & k & 2 & \ldots & 2 \\ \hline \end{array} \end{array}$$

Como os vetores de incidência dessas colorações estão em F_b, concluímos que

$$b_{i1} = b_{i2} = \ldots = b_{i,c-1} = b_{i,c+1} = \ldots = b_{ik}.$$

Denote por β_i esses coeficientes comuns (relativos ao vértice i e a todas as cores, a menos da cor c).

Assim, temos que b^T é da forma:

$$b^T = \begin{array}{|c|c|c|c|c|} \hline \beta_1 \ldots \beta_1 & \beta_2 \ldots \beta_2 & \ldots & \beta_i \ldots \beta_i \, b_{ic} \, \beta_i \ldots \beta_i & \ldots & \beta_k \ldots \beta_k \\ \hline \end{array}$$

.

Agora tome $\lambda_j := \beta_j$ para $j = 1, \ldots, n$, e $\alpha := b_{ic} - \beta_i$. É fácil ver que $b^T = \alpha a^T + \lambda^T M$. Resta provar que $\alpha \geq 0$. Para isto, tome as seguintes duas colorações convexas e seus respectivos vetores de incidência w_1 e w_2. (Lembramos que $c \neq 1$.)

$$C_1 = \begin{array}{|c|c|c|c|c|} \hline c & c & \ldots & c & c \\ \hline \end{array} \qquad w_1 = \begin{array}{|c|c|c|c|} \hline 00\ldots1\ldots00 & 00\ldots1\ldots00 & \ldots & 00\ldots1\ldots00 \\ \hline \end{array}$$

$$C_2 = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & \ldots & 1 & 1 \\ \hline \end{array} \qquad w_2 = \begin{array}{|c|c|c|c|} \hline 10\ldots0\ldots00 & 10\ldots0\ldots00 & \ldots & 10\ldots0\ldots00 \\ \hline \end{array}$$

Note que ambas as colorações são convexas, e portanto w_1 e w_2 são pontos do poliedro. O vetor w_2 pertence à face F_b (pois pertence a F_a). Então $b^T w_1 \geq b_0$ e $b^T w_2 = b_0$, e portanto,

$$\beta_1 + \ldots + \beta_{i-1} + b_{ic} + \beta_{i+1} + \ldots + \beta_n \geq b_0 \quad \text{e} \quad \beta_1 + \ldots + \beta_i + \ldots + \beta_n = b_0.$$

Dessas inequações concluímos que $b_{ic} \geq \beta_i$, e portanto $\alpha := b_{ic} - \beta_i \geq 0$. Logo, pela Proposição 5.4.2, temos que $x_{ic} \geq 0$ define uma faceta de $\mathcal{P}_{nk}$. □

Proposição 5.4.4 *Para todo vértice $i \in V$ e toda cor $c \in \mathcal{C}$, a inequação $x_{ic} \leq 1$ não define uma faceta de $\mathcal{P}_{nk}$.*

Prova Fixe $i \in V$ e $c \in \mathcal{C}$. Considere a seguinte equação válida para $\mathcal{P}_{nk}$.

$$x_{i1} + x_{i2} + \ldots + x_{ic} + \ldots + x_{ik} = 1. \tag{5.7}$$

Somando essa equação com as seguintes inequações de não-negatividade

$$-x_{ij} \leq 0 \quad \text{para} \quad j = 1, \ldots, n, \quad j \neq c,$$

obtemos a inequação $x_{ic} \leq 1$. Logo, esta inequação não define uma faceta de $\mathcal{P}_{nk}$. □

Vamos agora analisar as inequações de convexidade-generalizada do tipo t, onde $t \geq 1$. Lembramos que na seção anterior vimos que essas inequações são válidas para $\mathcal{P}_{nk}$.

Teorema 5.4.5 *Para toda cor $c \in \mathcal{C}$, a inequação $x_{pc} - x_{qc} + x_{rc} \leq 1$, onde $1 \leq p < q < r \leq n$, define uma faceta de $\mathcal{P}_{nk}$.*

Prova Fixe uma cor c e vértices p, q, r, tais que $1 \leq p < q < r \leq n$, e chame de $a^T x \leq a_0$ a inequação correspondente a $x_{pc} - x_{qc} + x_{rc} \leq 1$. Pelo Teorema 5.3.1, essa inequação é válida para $\mathcal{P}_{nk}$.

Seja $b^T x \leq b_0$ uma inequação válida para $\mathcal{P}_{nk}$ tal que

$$F_a := \{x \in \mathcal{P}_{nk} : a^T x = a_0\} \subseteq F_b := \{x \in \mathcal{P}_{nk} : b^T x = b_0\}.$$

Vamos mostrar que existe uma constante real $\alpha \geq 0$ e um vetor $\lambda \in \mathbb{R}^n$ tais que $b^T = \alpha a^T + \lambda^T M$.

Vejamos primeiramente a relação existente entre os coeficientes de b, onde

$$b^T = (b_{11}, \ldots, b_{1k}, \ldots, \ldots, b_{pc}, \ldots, \ldots, b_{qc}, \ldots, b_{rc}, \ldots, b_{n1}, \ldots, b_{nk}).$$

Tal como fizemos na prova do Teorema 5.4.3, vamos exibir colorações convenientes para obter o que desejamos. No que segue, f e d são cores em $\mathcal{C}$, distintas de c.

(a) Para $1 \leq i < p$ e $j \in \mathcal{C}$, defina C_j^i como segue:

$$C_j^i(v) = \begin{cases} f & \text{se} \quad 1 \leq v < i, \\ j & \text{se} \quad v = i, \\ c & \text{se} \quad i < v \leq n. \end{cases} \tag{5.8}$$

Tomando-se os vetores de incidência dessas colorações, e procedendo como na prova do Teorema 5.4.3, concluímos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para} \;\; 1 \leq i < p.$$

(b) Para $q < i < r$ e $j \in \mathcal{C}$, defina C_j^i como em (5.8).

Tomando-se os vetores de incidência dessas colorações, podemos concluir que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para } q < i < r.$$

(c) Para $p < i < q$ e $j \in \mathcal{C}$, defina C_j^i como segue:

$$C_j^i(v) = \begin{cases} c & \text{se} \quad v < i, \\ j & \text{se} \quad v = i, \\ f & \text{se} \quad v > i. \end{cases} \tag{5.9}$$

Tomando-se os vetores de incidência dessas colorações, concluímos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para} \;\; p < i < q.$$

(d) Para $i > r$, $j \in \mathcal{C}$, defina C_j^i como em (5.9).

Analogamente concluímos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para} \;\; r < i \leq n.$$

Vejamos agora como são os coeficientes b_{p*}, b_{q*} e b_{r*}.

Considere a coloração C_j^p para $j \in \mathcal{C} \setminus \{c\}$:

$$C_j^p(v) = \begin{cases} d & \text{se} \quad 1 \leq v < p, \\ j & \text{se} \quad v = p, \\ f & \text{se} \quad p < v < r, \\ c & \text{se} \quad r \leq v \leq n. \end{cases} \tag{5.10}$$

Claramente, os vetores de incidência dessas $k-1$ colorações pertencem a $F_a \subseteq F_b$, donde concluímos que os coeficientes de b relativos ao vértice p são todos idênticos, a menos do coeficiente b_{pc}. Denotamos por β_q esses termos idênticos.

Analogamente, considerando as colorações abaixo, C_j^q e C_j^r para $j \in \mathcal{C}\setminus\{c\}$, concluímos que os coeficientes de b relativos ao vértice q (resp. r) são todos idênticos, a menos dos coeficientes b_{qc} (resp. b_{rc}). Denotamos por β_q (resp. β_r) esses coeficientes idênticos.

$$C_j^q(v) = \begin{cases} c & \text{se} \quad 1 \leq v < q, \\ j & \text{se} \quad v = q, \\ f & \text{se} \quad q < v \leq n. \end{cases} \tag{5.11}$$

$$C_j^r(v) = \begin{cases} c & \text{se} \quad 1 \leq v < q, \\ d & \text{se} \quad q \leq v < r, \\ j & \text{se} \quad v = r, \\ f & \text{se} \quad r < v \leq n. \end{cases} \tag{5.12}$$

Em resumo, temos que b^T é da forma:

$\beta_1 \dots \beta_1$	$\dots$	$\beta_p \dots \beta_p \, b_{pc} \, \beta_p \dots \beta_p$	$\dots$	$\beta_q \dots \beta_q \, b_{qc} \, \beta_q \dots \beta_q$	$\dots$	$\beta_r \dots \beta_r \, b_{rc} \, \beta_r \dots \beta_r$	$\dots$

Considere agora as seguintes colorações convexas:

$$C_1 = (c, c, \dots, \overbrace{c}^{p}, c, \dots, \overbrace{c}^{q}, c, \dots, \overbrace{c}^{r}, \dots, c);$$

$$C_2 = (c, c, \dots, \overbrace{c}^{p}, f, \dots, \overbrace{f}^{q}, f, \dots, \overbrace{f}^{r}, \dots, f);$$

$$C_3 = (f, f, \dots, \overbrace{f}^{p}, f, \dots, \overbrace{f}^{q}, c, \dots, \overbrace{c}^{r}, \dots, c).$$

O vetor de incidência w_i da coloração C_i, para $i = 1, 2, 3$, pertence a F_a e portanto pertence a F_b. Logo, temos que $b^T w_i = b_0$.

De $b^T w_1 = b_0$ e $b^T w_3 = b_0$, segue que

$$b_{qc} = -b_{pc} + \beta_p + \beta_q. \tag{5.13}$$

De $b^T w_2 = b_0$ e $b^T w_3 = b_0$, temos que

$$b_{rc} = b_{pc} - \beta_p + \beta_r. \tag{5.14}$$

Sejam $\alpha := b_{pc} - \beta_p$ e $\lambda_i := \beta_i$ para $i = 1, \ldots, n$. Neste caso, temos que
$\alpha a^T + \lambda^T M = (\beta_1, \ldots, \beta_p, b_{pc}, \ldots, \beta_q, \underbrace{-b_{pc} + \beta_p + \beta_q}_{=b_{qc}}, \ldots, \beta_r, \underbrace{b_{pc} - \beta_p + \beta_r}_{=b_{rc}}, \ldots, \beta_n)$.

Usando as igualdades (5.13) e (5.14), concluímos que

$$b^T = \alpha a^T + \lambda^T M.$$

Para concluir a prova, resta provar que $\alpha := b_{pc} - \beta_p \geq 0$. Para ver isto, tome as seguintes colorações:

$$C' = (c, c, \ldots, \overbrace{c}^{p}, c, \ldots, \overbrace{c}^{q}, c, \ldots, \overbrace{c}^{r}, \ldots, c)$$

$$C'' = (f, f, \ldots, \overbrace{f}^{p}, f, \ldots, \overbrace{c}^{q}, c, \ldots, \overbrace{c}^{r}, \ldots, c).$$

Note que a coloração C' é convexa e seu vetor de incidência w' pertence a F_a, e portanto pertence a F_b, donde segue que $b^T w' = b_0$.

Já a coloração C'' é convexa, mas seu vetor de incidência w'' não pertence a F_a, porém pertence a $\mathcal{P}_{nk}$. Logo, $b^T w'' \leq b_0$, já que $b^T z \leq b_0$ é uma inequação válida para $\mathcal{P}_{nk}$. Da igualdade e da desigualdade mencionadas, temos que

$$b_{pc} + b_{qc} + b_{rc} = b_0 \geq \beta_p + b_{qc} + b_{rc},$$

donde segue que $b_{pc} \geq \beta_p$, e portanto $\alpha \geq 0$. Como queríamos demonstrar.

Com isso, pela Proposição 5.4.2 concluímos que a inequação $x_{pc} - x_{qc} + x_{rc} \leq 1$ define uma faceta de $\mathcal{P}_{nk}$. □

A seguir mostraremos que as inequações de convexidade-generalizada tipo t, onde $t \geq 2$, também definem facetas de $\mathcal{P}_{nk}$. A prova que faremos é praticamente uma generalização da prova que fizemos para as inequações tipo 1.

Teorema 5.4.6 *[Inequações de convexidade-generalizada do tipo t definem facetas]*
Para toda cor $c \in \mathcal{C}$, a inequação

$$\sum_{i=1}^{t} x_{p_i c} - \sum_{i=1}^{t} x_{q_i c} + x_{rc} \leq 1,$$

onde $1 \leq p = p_1 < q_1 < p_2 < q_2 < \ldots < p_t < q_t < r = p_{t+1} \leq n$, $t \geq 2$, define uma faceta de $\mathcal{P}_{nk}$.

Prova Fixe uma cor c e vértices $p, q_1, p_2, q_2, \ldots, p_t, q_t, r$, tais que $1 \leq p = p_1 < q_1 < p_2 < q_2 < \ldots < p_t < q_t < r \leq n$, $t \geq 2$. Seja $ax^T \leq a_0$ a inequação correspondente a $\sum_{i=1}^{t} x_{p_i c} - \sum_{i=1}^{t} x_{q_i c} + x_{rc} \leq 1$. Pelo Teorema 5.3.1, essa inequação é válida para $\mathcal{P}_{nk}$.

Seja $b^T \leq b_0$ uma inequação válida para $\mathcal{P}_{nk}$ tal que

$$F_a := \{x \in \mathcal{P}_{nk} : a^T x = a_0\} \subseteq F_b := \{x \in \mathcal{P}_{nk} : b^T x = b_0\}.$$

Vamos provar que existe uma constante real $\alpha \geq 0$ e um vetor $\lambda \in \mathbb{R}^n$ tais que $b^T = \alpha a^T + \lambda M$.

Estabeleceremos primeiro a relação existente entre os coeficientes de b, onde

$$b^T = (b_{11}, \ldots, b_{1k}, b_{21}, \ldots, b_{2k}, \ldots, b_{i1}, \ldots, b_{ic}, \ldots, b_{ik}, \ldots, b_{n1}, \ldots, b_{nk}).$$

Assim como foi feito no Teorema 5.4.5, tomamos colorações definidas como segue, cujos vetores de incidência pertencem a F_a e portanto a F_b. Dessa forma, obtemos em cada conjunto de colorações que definimos, uma relação entre os coeficientes de b^T. No que segue, d e f são cores em $\mathcal{C}$, distintas de c.

(a) Para $1 \leq i < p = p_1$ e $j \in \mathcal{C}$, considere:

$$C_j^i(v) = \begin{cases} f & \text{se } 1 \leq v < i, \\ j & \text{se } v = i, \\ c & \text{se } i < v \leq n. \end{cases}$$

Relacionando os vetores de incidência de cada uma dessas colorações, temos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para } 1 \leq i < p.$$

(b) Para $p_\ell < i < q_\ell$, onde $\ell = 1, \ldots, t$, e $j \in \mathcal{C}$, tome:

$$C_j^i(v) = \begin{cases} c & \text{se } 1 \leq v < i, \\ j & \text{se } v = i, \\ f & \text{se } i < v \leq n. \end{cases}$$

Novamente, relacionando os respectivos vetores de incidência dessas colorações, temos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para } p_\ell < i < q_\ell, \text{ onde } \ell = 1, \ldots, t.$$

(b1) Quando $i = q_\ell$, onde $\ell = 1, \ldots, t$, podemos tomar uma coloração como acima definida, considerando apenas o caso em que $j \in \mathcal{C} \setminus \{c\}$. Neste caso, concluímos que os coeficientes de b relativos ao vértice i são idênticos, a menos do coeficiente b_{ic}. Denotamos tais coeficientes por β_i.

(c) Para $q_\ell < i < p_{\ell+1}$, onde $\ell = 1, \ldots, t$, e $j \in \mathcal{C}$, considere a seguinte coloração:

$$C_j^i(v) = \begin{cases} f & \text{se } 1 \leq v < i, \\ j & \text{se } v = i, \\ c & \text{se } i < v \leq n. \end{cases}$$

Dos respectivos vetores de incidência dessas colorações, obtemos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para } q_\ell < i < p_{\ell+1}, \text{ onde } \ell = 1, \ldots, t.$$

(c1) Quando $i = p_\ell$, onde $\ell = 1, \ldots, t$, podemos tomar as seguintes colorações, considerando apenas o caso em que $j \in \mathcal{C} \setminus \{c\}$. Neste caso, concluímos que os coeficientes de b relativos ao vértice i são idênticos, a menos do coeficiente b_{ic}. Denotamos tais coeficientes por β_i.

$$C_j^i(v) = \begin{cases} f & \text{se } 1 \leq v < i, \\ j & \text{se } v = i, \\ d & \text{se } i < v < r, \\ c & \text{se } r \leq v \leq n. \end{cases}$$

(c2) Quando $i = r$, considere as seguintes colorações, onde $j \in \mathcal{C} \setminus \{c\}$.

$$C_j^i(v) = \begin{cases} c & \text{se } 1 \leq v \leq p_1, \\ f & \text{se } p_1 < v < r, \\ j & \text{se } v = r, \\ d & \text{se } r < v \leq n. \end{cases}$$

Neste caso, concluímos que os coeficientes de b correspondentes ao vértice r são todos idênticos, digamos β_r, a menos do coeficiente b_{rc}.

(d) Para $r < i \leq n$ e $j \in \mathcal{C}$, considere:

$$C_j^i(v) = \begin{cases} c & \text{se } 1 \leq v < i, \\ j & \text{se } v = i, \\ f & \text{se } i < v \leq n. \end{cases}$$

Relacionando os vetores de incidência das colorações acima, temos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para } r < i \leq n.$$

Em resumo, concluímos que

$$\beta_i := b_{i1} = b_{i2} = \ldots = b_{ik} \qquad \text{para } i \notin \{p_1, q_1, \ldots, p_t, p_{t+1}\}.$$

Também concluímos que os coeficientes de b, relativos ao vértice $i \in \{p_1, q_1, \ldots, p_t, p_{t+1}\}$ são todos idênticos, a menos dos coeficientes b_{ic}, e os denotamos por β_i.

Sejam

$$\alpha := b_{pc} - \beta_p, \quad \text{e}$$

$$\lambda_i := \beta_i \quad \text{para } i = 1, \ldots, n.$$

Afirmamos que $b^T = \alpha a^T + \lambda^T M$. Para provar isso, precisamos mostrar que

$$b_{ic} = \begin{cases} +b_{pc} - \beta_p + \beta_i & \text{se } i \in \{p_2, \ldots, p_{t+1}\}, \\ -b_{pc} + \beta_p + \beta_i & \text{se } i \in \{q_1, q_2, \ldots, q_t\}. \end{cases} \tag{5.15}$$

Como a ideia da prova é análoga àquelas vistas anteriormente, vamos apenas indicar as colorações convexas que produzirão as igualdades desejadas.

Considere as $t+1$ colorações convexas $C_0, C_1, \ldots, C_t$, onde

$$C_0(v) = c \qquad \text{para} \;\; v = 1, \ldots, n.$$

Para $\ell = 1, \ldots, t$, seja

$$C_\ell(v) = \begin{cases} f & \text{se} \;\; v \leq q_\ell, \\ c & \text{caso contrário.} \end{cases}$$

Comparando apropriadamente os vetores de incidência dessas colorações, que pertencem a F_b, podemos concluir que

$$b_{p_\ell c} + b_{q_\ell c} = \beta_{p_\ell} + \beta_{q_\ell} \qquad \text{para} \;\; \ell = 1, \ldots, t. \tag{5.16}$$

Considere agora as seguintes t colorações convexas $C'_1, C'_2, \ldots, C'_t$.
Para $\ell = 1, \ldots, t$, seja

$$C'_\ell(v) = \begin{cases} c & \text{se} \;\; v < q_\ell, \\ f & \text{caso contrário.} \end{cases}$$

Comparando os vetores de incidência dessas colorações, concluímos que

$$b_{q_\ell c} + b_{p_{\ell+1} c} = \beta_{q_\ell} + \beta_{p_{\ell+1}} \qquad \text{para} \;\; \ell = 1, \ldots, t. \tag{5.17}$$

Combinando as igualdades (5.16) e (5.17), podemos provar a relação dada em (5.15). Com isso, concluímos que

$$b^T = \alpha a^T + \lambda^T M.$$

Resta agora provar que $\alpha := b_{pc} - \beta_p \geq 0$. Para isso, basta comparar os vetores de incidência das seguintes colorações: C_0 (já definida antes) e C''.

$$C_0(v) = c \quad \text{para} \;\; v = 1, \ldots, n.$$

$$C''(v) = \begin{cases} f & \text{se} \;\; v \leq p, \\ c & \text{caso contrário.} \end{cases}$$

Note que a coloração C_0 é convexa e seu vetor de incidência w_0 pertence $F_a \subseteq F_b$. Logo, $b^T w_0 = b_0$. Já a coloração C'' é convexa e seu vetor de incidência w'' não pertence a F_a mas pertence a $\mathcal{P}_{nk}$. Logo, $b^T w'' \leq b_0$. Portanto, $b^T w_0 \geq b^T w''$, donde segue que

$b_{pc} \geq \beta_p$, ou seja, $\alpha := b_{pc} - \beta_p \geq 0$.

Concluímos assim que as inequação de convexidade-generalizada do tipo t, onde $t \geq 2$, definem facetas de $\mathcal{P}_{nk}$. □

No próximo capítulo discutiremos um algoritmo polinomial que resolve o problema da separação para a classe de inequações de convexidade-generalizada do tipo t.

Capítulo 6

Algoritmos e resultados computacionais para o problema RCC

Neste capítulo apresentamos os resultados computacionais obtidos com os algoritmos que implementamos para o problema RCC.

Começamos apresentando os primeiros resultados que obtivemos com uma heurística simples e gulosa, para ter uma ideia da qualidade da relaxação linear que consideramos. Comparamos o desempenho dessa heurística com o do algoritmo de 2-aproximação de Moran e Snir.

A seguir, apresentamos os resultados computacionais de um algoritmo *branch-and-cut* que implementamos para o problema RCC, usando os resultados sobre o poliedro $\mathcal{P}_{nk}$, obtidos no capítulo anterior. Apresentamos também os resultados computacionais obtidos ao considerar uma relaxação do poliedro $\mathcal{P}_{nk}$, que trataremos neste capítulo.

Veremos que as inequações de convexidade-generalizada têm um papel fundamental no algoritmo *branch-and-cut* que desenvolvemos. Descrevemos aqui um algoritmo de programação dinâmica que resolve em tempo polinomial o problema da separação para essa classe de inequações. Esse algoritmo é a alma do algoritmo *branch-and-cut*, tendo se mostrado muito importante para o bom desempenho desse algoritmo.

Para o problema 2-RCC (caso especial de caminhos em que cada cor ocorre no máximo duas vezes), apresentamos resultados computacionais comparando o algoritmo de 3/2-aproximação e o algoritmo *branch-and-cut* para este tipo especial de caminho.

6.1 Preliminares

Para todos os resultados computacionais, a menos do algoritmo de 2-aproximação, utilizamos um *software* de otimização chamado *Gurobi Optimization*. Trata-se de um pacote de alta performance para resolver problemas de programação linear, programação quadrática e inteira-mista.

Todos os testes deste capítulo foram realizados em uma máquina 64 bits, com dois processadores Intel® Xeon® *E*5440 (*quad-core*), com velocidade de *clock* 2.83*GHz*, 32GB de RAM.

Todas as instâncias usadas para os testes foram geradas aleatoriamente. Cada instância aqui é representada por uma sequência de números que é obtida da seguinte forma: para cada vértice de um caminho, sorteamos um número de 1 a k que representa uma cor, e atribuímos esta cor ao vértice.

6.2 Uma heurística gulosa

Implementamos uma heurística simplesmente baseada na formulação linear 0/1 do problema RCC, que chamaremos de heurística **H**.

A ideia dessa heurística consiste em resolver a relaxação linear do PL/01. Se a solução obtida é 0/1, então essa solução é ótima, e a heurística devolve essa solução. Se não, procuramos uma variável cujo valor fracionário seja maior ou igual t, onde t é inicialmente 0.9. O valor desta variável é arredondado para 1 e o processo é repetido. O valor de t decresce para $t - 0.1$ se, para a solução atual não é encontrada nenhuma variável com valor fracionário maior ou igual a t. O processo é repetido, se necessário, até que tenhamos $t = 0.5$.

Para compararmos o desempenho desta heurística com o desempenho do algoritmo de 2-aproximação de Moran e Snir[16], apresentado no Capítulo 4, implementamos a versão sem pesos desse último algoritmo. Chamamos de **A** esse algoritmo, que na sua versão com pesos é chamado de AproxRCC-MS-pesos (veja Seção 4.3.1).

Ressaltamos aqui que, no Capítulo 5, a formulação linear considerada é um problema de maximização. Como nessa formulação contabilizamos o número de cores que são mantidas, o valor da solução do RCC (problema de minimização) é obtida subtraindo esse valor de n.

Descrevemos a seguir as tabelas 6.1, 6.2 e 6.3. Nessas tabelas consideramos instâncias (P, C), onde P tem n vértices e $|\mathcal{C}| = k$.

- Para cada par (n, k), geramos 10 instâncias, e exibimos os resultados para cada uma delas. Note que o valor de k está indicado no topo de cada grupo de 10 instâncias, e o valor de n corresponde aos valores indicados em cada linha.

- Nas colunas 0/1, **Alg A**, **frac**, e **heur** estão indicados os valores das soluções obtidas com o PL/01 (solução ótima), o algoritmo de 2-aproximação, a relaxação linear e a heurística H que descrevemos. Para cada caso também indicamos o tempo gasto (em minutos).

- Nas últimas duas colunas de cada tabela exibimos a razão entre o valor da solução da heurística H (resp. algoritmo A) e o valor da solução 0/1 (ótima).

Comparando a qualidade das soluções encontradas pela heurística H e pelo algoritmo A (visível nas duas últimas colunas), verificamos que ela é boa em ambos os casos.

Quanto ao tempo gasto, observamos que, à medida que o número de vértices aumenta, o tempo gasto pela heurística H é bem maior que o tempo gasto pelo algoritmo A.

No geral, tanto a heurística H quanto o algoritmo A encontraram soluções que são no máximo 1.2 vezes maior que o valor da solução ótima.

Observamos também que para resolver o PL/01 para instâncias com 70 vértices foi gasto uma média de 22 horas, e em algumas delas, o tempo gasto foi de até 78 horas.

5 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
30	16	0,09	19	12,25	19	0,02	1,19	1,19
30	11	0,01	13	11	11	0,01	1,00	1,18
30	17	0,25	22	12,75	19	0,02	1,12	1,29
30	16	0,03	18	13	16	0,01	1,00	1,13
30	14	0,02	14	13	17	0,02	1,21	1,00
30	13	0,02	13	12,25	16	0,02	1,23	1,00
30	15	0,04	16	13	18	0,02	1,20	1,07
30	12	0,01	12	11,34	12	0,02	1,00	1,00
30	15	0,07	17	12	15	0,01	1,00	1,13
30	15	0,03	17	13,01	16	0,02	1,07	1,13
média		**0,06**				**0,02**	**1,10**	**1,11**

15 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
30	13	0,07	16	11,6	15	0,04	1,15	1,23
30	12	0,05	14	11,13	12	0,04	1,00	1,17
30	12	0,07	14	11	13	0,03	1,08	1,17
30	11	0,05	13	10,5	11	0,03	1,00	1,18
30	12	0,05	15	10,5	12	0,03	1,00	1,25
30	13	0,06	17	12	14	0,03	1,08	1,31
30	12	0,06	14	10,5	15	0,04	1,25	1,17
30	12	0,05	14	11,25	14	0,05	1,17	1,17
30	11	0,05	12	10,5	11	0,03	1,00	1,09
30	13	0,05	16	12,17	13	0,04	1,00	1,23
média		**0,06**				**0,04**	**1,07**	**1,20**

25 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
30	9	0,08	10	9	9	0,05	1,00	1,11
30	10	0,21	11	8,5	11	0,06	1,10	1,10
30	11	0,09	11	9,5	13	0,07	1,18	1,00
30	11	0,09	11	8	11	0,06	1,00	1,00
30	10	0,09	10	9	12	0,05	1,20	1,00
30	8	0,08	10	7,5	8	0,04	1,00	1,25
30	9	0,07	11	9	12	0,04	1,33	1,22
30	11	0,09	12	9,25	11	0,05	1,00	1,09
30	12	0,08	13	10,5	12	0,05	1,00	1,08
30	9	0,08	11	8,75	9	0,05	1,00	1,22
média		**0,10**				**0,05**	**1,08**	**1,11**

Tabela 6.1: *Soluções do PL/01, do algoritmo A e da heurística H, para $n = 30$.*

10 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
50	30	11,85	35	22,75	34	0,43	1,13	1,17
50	29	7,57	36	22,75	33	0,70	1,14	1,24
50	26	1,51	28	21,5	29	0,19	1,12	1,08
50	26	1,13	32	21,9	28	0,30	1,08	1,23
50	27	0,74	30	22,45	30	0,47	1,11	1,11
50	27	3,79	33	22,25	32	0,40	1,19	1,22
50	28	4,12	37	22	30	0,28	1,07	1,32
50	26	0,74	32	21,5	28	0,25	1,08	1,23
50	29	4,28	34	22,5	32	0,41	1,10	1,17
50	28	4,81	32	22,25	30	0,28	1,07	1,14
média		**4,05**				**0,37**	**1,11**	**1,19**

25 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
50	23	2,12	24	20,5	24	1,11	1,04	1,04
50	19	1,11	22	17,38	21	0,58	1,11	1,16
50	23	1,20	25	19,75	26	0,72	1,13	1,09
50	23	1,32	27	20,75	25	0,86	1,09	1,17
50	23	1,79	24	18,25	24	0,63	1,04	1,04
50	27	20,49	29	22	31	0,88	1,15	1,07
50	22	3,64	22	19,25	22	0,90	1,00	1,00
50	23	1,36	29	20,63	23	0,60	1,00	1,26
50	22	1,06	25	18,75	25	0,65	1,14	1,14
50	24	5,25	27	19,93	25	0,63	1,04	1,13
média		**3,93**				**0,75**	**1,07**	**1,11**

40 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
50	15	1,40	16	13	17	0,39	1,13	1,07
50	20	1,70	21	19	22	0,72	1,10	1,05
50	19	1,53	21	16,75	20	0,80	1,05	1,11
50	19	2,72	22	18	23	1,55	1,21	1,16
50	21	2,18	21	19	24	1,29	1,14	1,00
50	17	1,25	20	16	17	0,44	1,00	1,18
50	15	1,23	18	14,25	15	0,53	1,00	1,20
50	21	1,44	22	17,75	23	0,80	1,10	1,05
50	21	1,33	21	18,5	23	0,58	1,10	1,00
50	16	1,22	18	16	18	0,61	1,13	1,13
média		**1,60**				**0,77**	**1,10**	**1,09**

Tabela 6.2: *Soluções do PL/*01*, do algoritmo A e da heurística H, para* $n = 50$.

10 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
70	45	2.110,6	51	32	51	7,8	1,13	1,13
70	42	250,8	49	32	48	5,0	1,14	1,17
70	46	3.487,8	51	32,5	51	5,2	1,11	1,11
70	41	85,0	50	31	45	2,6	1,10	1,22
70	44	1.478,2	49	31	50	1,6	1,14	1,11
70	46	3.646,1	53	32,25	54	5,8	1,17	1,15
70	45	4.188,4	50	31,75	51	12,2	1,13	1,11
70	44	1.096,0	53	32,5	49	17,8	1,11	1,20
70	43	420,8	50	32	51	4,1	1,19	1,16
70	42	199,1	53	32,25	43	11,7	1,02	1,26
média		**1.650,2**				**7,4**	**1,12**	**1,16**

20 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
70	40	186,1	47	32,25	45	6,3	1,13	1,18
70	40	110,0	48	32,5	46	8,2	1,15	1,20
70	39	356,0	43	31,25	44	5,0	1,13	1,10
70	39	262,9	44	31,38	44	2,8	1,13	1,13
70	39	191,5	45	31,5	48	3,0	1,23	1,15
70	42	658,4	48	32,75	50	4,1	1,19	1,14
70	42	2.782,3	49	32,25	43	3,6	1,02	1,17
70	40	471,0	42	30,75	40	8,1	1,00	1,05
70	39	422,0	42	31,25	41	5,2	1,05	1,08
70	41	353,4	46	32,5	49	3,8	1,20	1,12
média		**579,4**				**5,0**	**1,12**	**1,13**

35 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
70	37	378,8	39	30,42	41	9,5	1,11	1,05
70	34	23,9	37	28,5	41	10,6	1,21	1,09
70	28	44,0	31	25,75	32	7,1	1,14	1,11
70	32	33,7	38	27,5	35	8,0	1,09	1,19
70	32	37,2	35	29	34	3,9	1,06	1,09
70	31	246,1	34	25,5	35	9,6	1,13	1,10
70	35	198,4	37	28,5	42	7,9	1,20	1,06
70	32	111,4	35	27,5	35	3,4	1,09	1,09
70	37	70,9	39	30	42	4,8	1,14	1,05
70	33	91,1	35	27,75	41	4,8	1,24	1,06
média		**123,6**				**7,0**	**1,14**	**1,09**

50 cores								
n	0/1		Alg A	frac	heur		H/01	A/01
	valor	tempo(m)	valor	valor	valor	tempo(m)		
70	30	4.695,6	31	25,25	32	349,1	1,07	1,03
70	29	1.844,3	31	25,75	31	394,7	1,07	1,07
70	28	2.260,6	31	23,75	31	353,1	1,11	1,11
70	31	3.876,3	33	26,25	34	469,0	1,10	1,06
70	28	1.802,3	30	25	33	285,3	1,18	1,07
70	30	2.910,7	32	26,67	35	444,3	1,17	1,07
70	31	3.948,6	35	27,5	33	294,2	1,06	1,13
70	29	1.546,3	30	24	31	315,7	1,07	1,03
70	28	3.281,6	30	24,01	34	484,7	1,21	1,07
70	27	1.474,6	30	23,75	32	493,4	1,19	1,11
média		**2.764,1**				**388,4**	**1,12**	**1,08**

Tabela 6.3: *Soluções do PL/01, do algoritmo A e da heurística H, para $n = 70$.*

6.3 Um algoritmo *branch-and-cut* para o problema RCC

Os resultados que apresentamos nas tabelas anteriores (Tabela 6.1, Tabela 6.2 e Tabela 6.3) nos motivaram a investigar a fundo o poliedro $\mathcal{P}_{nk}$ que associamos ao problema RCC.

Basicamente, procuramos encontrar inequações que definem facetas do poliedro $\mathcal{P}_{nk}$, para explorá-las num esquema de *branch-and-cut*. Para que essas inequações sejam úteis do ponto de vista algorítmico, precisamos de algoritmos de separação para essas inequações, que sejam eficientes. Veremos a seguir que para uma classe bem grande de inequações, temos um algoritmo de separação polinomial. Descrevemos esse algoritmo na Seção 6.3.3.

6.3.1 Ideia básica do método *branch-and-cut*

No capítulo anterior estudamos o poliedro $\mathcal{P}_{nk}$, com vistas ao desenvolvimento de um algoritmo *branch-and-cut* para o problema RCC. Como este problema é NP-difícil, sabemos que não é possível encontrar uma descrição completa e boa do poliedro $\mathcal{P}_{nk}$, a menos que P = NP. No entanto, ao encontrar facetas desse poliedro, obtemos uma formulação linear mais forte que pode ser útil do ponto-de-vista prático, quando usado num algoritmo *branch-and-cut*.

O método *branch-and-cut* combina o método *branch-and-bound* — que enumera todas as soluções viáveis do problema em uma estrutura de árvore de decisão — com o uso de planos-de-corte (*cuts*). Estes últimos são encontrados pelos algoritmos de separação que são desenvolvidos para classes de inequações válidas do poliedro em foco. Um plano-de-corte nada mais é do que uma inequação válida para o poliedro que não é satisfeita pela solução da relaxação linear corrente.

A idéia deste método consiste em resolver a relaxação linear do modelo, ou seja remover as condições de integralidade e usar este valor obtido, que chamamos de limitante dual, para realizar podas por limitantes na árvore. De fato, se em um nó da árvore de *branch-and-bound*, a relaxação for menor ou igual à melhor solução conhecida até agora, não precisamos explorar a subárvore definida por esse nó, uma vez que o limitante dual é sempre maior ou igual ao valor de qualquer solução obtida nessa subárvore.

No entanto, à medida que exploramos tal árvore de decisão, o número de problemas a serem resolvidos tende a crescer muito. O que podemos fazer, por exemplo, é em cada nó desta árvore de *branch-and-bound* encontrar novas inequações válidas para o poliedro, que estão sendo violadas pela solução relaxada atual. Isto implica em eliminar

soluções fracionárias, de modo que a cada iteração tenhamos um melhor limitante superior para o problema, e estejamos mais próximos de uma solução inteira.

6.3.2 Notação

Para facilitar o entendimento dos próximos resultados, representaremos aqui um vetor $x = (x_{ic} : 1 \leq i \leq n, 1 \leq c \leq k) \in \mathbb{R}^{nk}$ por uma matriz $\Gamma(x) \in \mathbb{R}^{n \times k}$, onde

$$\Gamma(x) = (\gamma_{ic})_{\substack{1 \leq i \leq n \\ 1 \leq c \leq k}} \quad \text{é tal que} \quad \gamma_{ic} = x_{ic}.$$

Assim, as linhas da matriz $\Gamma(x)$ representam os vértices $i = 1, \ldots, n$ (do caminho P), e as colunas representam as cores $c = 1, \ldots, k$.

Quando $x \in \{0,1\}^{nk}$, então x representa uma coloração de $P = (1, \ldots, n)$. Neste caso, para verificar se x representa uma coloração convexa, basta verificar se $\Gamma(x)$ tem as seguintes propriedades:

- Cada linha de $\Gamma(x)$ deve ter exatamente uma entrada 1. Isto equivale a exigir que $\sum_{c=1}^{k} x_{ic} = 1$, para cada linha i.
- Em cada coluna de $\Gamma(x)$, as entradas iguais a 1 (se houver) devem ser consecutivas. Isto equivale a observar que para cada coluna c, $1 \leq c \leq n$, temos que a restrição $x_{pc} - x_{qc} + x_{rc} \leq 1$ deve estar satisfeita para todo p, q e r tais que $1 \leq p < q < r \leq n$.

No exemplo a seguir (Figura 6.1), temos duas matrizes $\Gamma' = \Gamma(x')$ e $\Gamma'' = \Gamma(x'')$, ambas em $\{0,1\}^{9 \times 5}$, que definem duas colorações distintas para um caminho $P = (1, \ldots, 9)$.

Observe que somente a coloração representada por x' é convexa. A coloração representada por x'' não é convexa: note que a coluna 3 não tem a propriedade dos 1´s consecutivos, e de fato, a restrição $x_{23} - x_{43} + x_{63} \leq 1$ não está satisfeita.

6.3.3 Algoritmo de separação para algumas facetas do poliedro $\mathcal{P}_{nk}$

Seja c uma cor em $\mathcal{C}$. Considere a inequação abaixo, relativa a c, que provamos definir uma faceta do poliedro $\mathcal{P}_{nk}$,

$$\sum_{i=1}^{t} x_{p_i c} - \sum_{i=1}^{t} x_{q_i c} + x_{rc} \leq 1, \tag{6.1}$$

onde p, r, t são inteiros tais que $1 \leq p < r - 1 < n$, $1 \leq t \leq \lfloor \frac{r-p}{2} \rfloor$ e p_i e q_i são inteiros tais que $p = p_1 < q_1 < p_2 < q_2 < \ldots < p_t < q_t < r$.

$$\Gamma' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad \Gamma'' = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Figura 6.1: *Somente a matriz* Γ' *define uma coloração convexa de* $P = (1, \ldots, 9)$.

Seja x uma solução do PL relaxado. Se x é uma solução inteira, então x é uma solução ótima. Se x não é uma solução inteira, queremos encontrar uma inequação válida para $\mathcal{P}_{nk}$ que seja violada por x. Descrevemos a seguir um algoritmo — chamado **Separação-de-Facetas** — que, para uma dada solução x, verifica se existe uma inequação como acima descrita que seja violada por x.

Este algoritmo recebe como entrada uma coluna, digamos c, da matriz $\Gamma(x)$, que denotamos aqui por $v \in \mathbb{R}^n$. Ou seja, $v = (v_1, \ldots, v_n)$ é tal que $v_i = x_{ic}$.

Para este vetor v, o algoritmo encontra uma expressão do tipo abaixo cujo valor é o maior possível. (Note que t pode variar de 1 a $\lfloor (r-p)/2 \rfloor$.)

$$\sum_{i=1}^{t} v_{p_i} - \sum_{i=1}^{t} v_{q_i} + v_r, \tag{6.2}$$

$$\text{onde } p = p_1 < q_1 < p_2 < q_2 < \ldots < p_t < q_t < r.$$

Vejamos como podemos resolver isso. Sejam

$$\text{Soma}(p, r) := \max_{1 \le t \le \lfloor \frac{r-p}{2} \rfloor} \left\{ \sum_{i=1}^{t} v_{p_i} - \sum_{i=1}^{t} v_{q_i} + v_r \right\}.$$

$$S(p) := \max_{p+1 < r \le n} \text{Soma}(p, r).$$

$$\text{ValMax}(c) := \max_{1 \le p < n-1} S(p).$$

Podemos calcular $S(p)$ através da seguinte fórmula de recorrência:

$$S(n) = v_n; \quad S(n-1) = v_{n-1};$$

$$S(p) = \max_{p+2 \leq r \leq n} \left\{ v_p,\ v_p + \max_{p<q<r} \left\{ -v_q + S(r) \right\} \right\}.$$

Um algoritmo de programação dinâmica pode ser implementado para calcular $S(p)$, e assim obtermos o valor de ValMax(c).

Se ValMax(c) ≤ 1, o processo é repetido para uma nova cor. Se ValMax(c) > 1, então a solução x viola a correspondente inequação relativa à cor c. Na implementação que fizemos, cada vez que detectamos uma cor c para a qual ocorre violação, acrescentamos a inequação correspondente ao sistema linear corrente, e voltamos a resolver o PL com essa restrição adicional.

Para sabermos qual é a inequação correspondente, o algoritmo deve ser implementado de forma a armazenar os índices $p = p_1, q_1, \ldots, p_t, q_t, r$ que definem $S(p)$. À medida que melhores $S(p)$ são encontrados, basta armazenar apenas os índices do melhor $S(p)$ até o momento em análise.

Para armazenar esses índices, basta usar um vetor binário, digamos $bin\,(1, \ldots, n)$, que recebe apenas 1 nas posições correspondentes aos índices $p = p_1, q_1, \ldots, p_t, q_t, r$.

Dessa forma, a inequação correspondente (que deve ser acrescentada por violar a solução corrente) pode ser obtida simplesmente varrendo o vetor bin, e alternando os sinais + e − para as posições que contêm 1:

$$x_{p_1,c} - x_{q_1,c} + x_{p_2,c} - x_{q_2,c} + \ldots + x_{p_t,c} - x_{q_t,c} + x_r \leq 1.$$

Por exemplo, para $n = 10$, se $bin = (0, 1, 1, 0, 0, 1, 0, 1, 1, 0)$ é o vetor binário devolvido pelo algoritmo para uma coluna c em que $S^*(c) > 1$, então a restrição correspondente é

$$x_{2c} - x_{3c} + x_{6c} - x_{8c} + x_{9c} \leq 1.$$

Observamos que, em vez de inserirmos uma inequação violada a cada vez que uma tal inequação é encontrada, podemos inserir mais do que uma de cada vez. Independente da heurística envolvida, este processo é repetido até que não seja mais encontrada uma restrição violada pela solução atual.

Na Figura 6.2 damos exemplo de uma solução fracionária x, que é uma solução ótima para o PL relaxado, e exibimos a inequação violada por x que é encontrada pelo algoritmo de separação que descrevemos.

6.3.4 Descrição do algoritmo para o problema RCC

Descrevemos a seguir detalhes da implementação do **AlgoritmoRCC**, que propomos para resolver o problema RCC. Este algoritmo tem como subrotina o algoritmo

$$\Gamma = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0.1 & 0.2 & 0.4 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.1 & 0.2 & 0.4 & 0.1 & 0.2 \end{pmatrix} \end{array}$$

Figura 6.2: *A seguinte restrição* $x_{13} - x_{23} + x_{33} - x_{43} + x_{53} - x_{63} + x_{73} - x_{83} + x_{93} \leq 1$ *não está satisfeita.*

Separação-de-Facetas que é o algoritmo de separação mencionado na seção anterior.

Primeiramente, vale observar que adotamos no AlgoritmoRCC um limite superior e um limite inferior para o valor da solução inteira ótima. De fato, consideramos como limite inferior o valor da solução devolvida pelo algoritmo de 2-aproximação. Já o limite superior considerado é o valor da solução ótima do PL relaxado. Além disso, consideramos duas fases na execução do AlgoritmoRCC que descrevemos a seguir.

Fase 1 A partir da solução ótima do PL relaxado, que é uma solução fracionária, o algoritmo Separação-de-Facetas encontra e adiciona à formulação linear corrente alguma restrição do tipo 6.2 que está sendo violada por esta solução relaxada. A formulação corrente é reotimizada. Esse processo é repetido até que seja encontrada uma solução em que não existem mais restrições desta classe. Com sorte, esta solução pode ser inteira, e dessa forma, a solução inteira ótima para o problema é obtida.

Fase 2 Caso esta solução não seja inteira, sabemos que, pelo menos, o valor da solução devolvida na primeira fase nos dá um novo limite superior para o valor da solução ótima, sendo assim, o valor do limite superior é atualizado no AlgoritmoRCC. Nesse momento, em cada nó do *branch-and-bound* o algoritmo procura na atual solução fracionária se existem restrições do tipo 6.2 que ainda estão sendo violadas e, neste caso, novas restrições são adicionadas à formulação corrente. O AlgoritmoRCC é finalizado se, em algum nó do *branch-and-bound* é encontrada uma solução inteira cujo valor é igual ao limite superior, ou se todos os nós da árvore de *branch-and-bound* já foram visitados.

6.3.5 Resultados experimentais preliminares do AlgoritmoRCC

Ressaltamos que outras estratégias podem ser adotadas na primeira e na segunda fase. Em alguns casos, por exemplo, a primeira fase pode levar bastante tempo, assim uma saída é parar esta fase, caso ultrapasse um certo tempo determinado, e executar a segunda fase. Uma outra tática que podemos tomar, referente à segunda fase, consiste em restringir a quantidade de novas inequações que são adicionadas em cada nó do *branch-and-bound*.

Em uma árvore de *branch-and-bound*, em cada nó, temos um problema a ser otimizado com uma restrição a mais de alguma variável fixada. Assim, à medida que solucionamos um nó, novos problemas são criados. Quanto à ordem que os problemas são resolvidos, podemos escolher se desejamos percorrer os nós em largura ou em profundidade na árvore de *branch-and-bound*.

Para os resultados que apresentamos nas tabelas 6.4, 6.5, consideramos que os nós são percorridos em largura. A menos disso, não consideramos nenhuma estratégia a mais, relativamente às que já descrevemos na primeira e na segunda fase.

Apresentamos a seguir uma descrição das tabelas 6.4, 6.5.

- Lembramos que todas as instâncias usadas para os experimentos foram geradas aleatoriamente, conforme explicamos anteriormente. Em especial, cada uma das instâncias geradas para os próximos experimentos está associada a uma semente, de modo que, sempre que necessário, ela pode ser recuperada.

- Para cada par (n, k) geramos 10 instâncias; sendo que o número de cores k foi tomado considerando uma porcentagem aproximada de 20%, 50% e 70% sobre o número de vértices. Note que em cada par (n, k) estamos trabalhando com um problema de $n \times k$ variáveis.

- Na segunda coluna de cada tabela, temos o valor da solução do algoritmo de 2-aproximação para cada uma das instâncias, que denotamos por **A** (vale lembrar que o algoritmo A é o mesmo algoritmo AproxRCC-MS-Pesos apresentado no Capítulo 4, só que na versão sem pesos).

- Na terceira e quarta colunas temos o valor e o tempo (em minutos) da solução ótima devolvido pelo AlgoritmoRCC.

- A penúltima coluna indica se a instância precisou entrar no método *branch-and-bound* mencionado na descrição do AlgoritmoRCC. A resposta "Não" indica que obtivemos solução inteira ótima já na primeira fase.

- Na última coluna apresentamos a razão entre o valor da solução devolvida pelo algoritmo de 2-aproximação (Alg A) e o valor solução ótima obtida pelo AlgoritmoRCC.

A seguir, descrevemos algumas conclusões que obtivemos com a implementação do AlgoritmoRCC e o que observamos nos resultados que apresentamos nas tabelas 6.4, 6.5 e nas Figuras 6.3 e 6.4.

▹ O pacote *Gurobi Optimizer* possui, dentre seus processos, um algoritmo inicial chamado de *presolve*. Este algoritmo entra em ação sempre que passamos um modelo para ser otimizado. A função do *presolve* é simplificar o modelo removendo dados desnecessários. Com isso temos a informação do tamanho do modelo que será otimizado depois do *presolve*. Observamos que para instâncias com 100 vértices e 70 cores, por exemplo, o tamanho do modelo a ser otimizado possui 7000 linhas, 2.836.850 colunas e 8.503.250 coeficientes não-nulos. Já para instâncias de 50 vértices e 35 cores, o tamanho do modelo resultante possui 1750 linhas, 687.800 colunas e 2.061.500 coeficientes não-nulos.

▹ Observando as tabelas 6.4, 6.5, concluímos que o AlgoritmoRCC devolve solução ótima para instâncias de até 100 vértices em no máximo 4 horas. Vale lembrar que para obtermos solução do PL0/1, o tempo gasto foi de até 78 horas para instâncias com 70 vértices.

▹ Nas Figuras 6.3 e 6.4 apresentamos gráficos que comparam a quantidade de trocas de cores feita pelo algoritmo de 2-aproximação com a quantidade de trocas de cores feitas pelo AlgortimoRCC. Os gráficos foram construídos com base nos dados obtidos nas respectivas tabelas 6.4, 6.5. Mais especificamente, para cada par (n, k), temos um gráfico associado. Vale ressaltar, baseando-se em todos resultados experimentais em que utilizamos o algoritmo de 2-aproximação, que o valor da solução obtida deste algoritmo não divergiu, em média, mais do que 20% da solução inteira ótima, o que demonstra que, o algoritmo de 2-aproximação possui uma excelente performance em termos de qualidade de solução.

▹ Notamos que, para todas as instâncias, sem exceção, que necessitaram da segunda fase do AlgoritmoRCC, o valor da solução fracionária obtida na primeira fase do AlgoritmoRCC foi o mesmo valor da solução ótima inteira devolvida no final do método *branch-and-cut*. Isso mostra que as inequações que definem facetas nos deixam sempre muito perto de uma solução inteira ótima; e no pior dos casos, em que a solução inteira não é encontrada na primeira fase, o uso destas inequações nos fornece um excelente limitante superior para o problema ainda a ser resolvido.

5 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
50	34	30	0,21	Sim	1,13
50	38	30	0,11	Não	1,27
50	31	29	0,08	Não	1,07
50	26	26	0,08	Não	1,00
50	29	26	0,07	Não	1,12
50	35	28	0,12	Sim	1,25
50	37	31	0,12	Não	1,19
50	27	25	0,04	Não	1,08
50	37	30	0,13	Não	1,23
50	31	27	0,06	Não	1,15
média			**0,10**		**1,15**

10 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
50	31	28	0,60	Não	1,11
50	35	30	0,74	Sim	1,17
50	37	29	0,21	Não	1,28
50	36	27	0,14	Não	1,33
50	35	30	2,25	Sim	1,17
50	33	28	0,16	Não	1,18
50	32	29	0,25	Não	1,10
50	33	30	0,26	Não	1,10
50	33	30	1,02	Sim	1,10
50	32	27	0,14	Não	1,19
média			**0,58**		**1,17**

25 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
50	22	21	0,87	Não	1,05
50	27	24	1,25	Não	1,13
50	24	22	1,55	Não	1,09
50	25	23	3,67	Sim	1,09
50	26	23	1,10	Não	1,13
50	26	22	0,37	Sim	1,18
50	26	22	0,19	Sim	1,18
50	26	24	0,88	Sim	1,08
50	24	22	0,88	Não	1,09
50	28	25	1,60	Não	1,12
média			**1,24**		**1,11**

35 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
50	22	19	4,38	Sim	1,16
50	19	19	3,32	Sim	1,00
50	21	20	1,87	Não	1,05
50	23	21	2,29	Não	1,10
50	20	19	1,44	Sim	1,05
50	22	20	4,67	Não	1,10
50	18	18	0,87	Não	1,00
50	22	20	3,05	Sim	1,10
50	19	19	0,66	Não	1,00
50	21	18	0,89	Não	1,17
média			**2,34**		**1,07**

Tabela 6.4: *Algoritmo de* 2*-aproximação* $\times$ *AlgoritmoRCC para* $n = 50$.

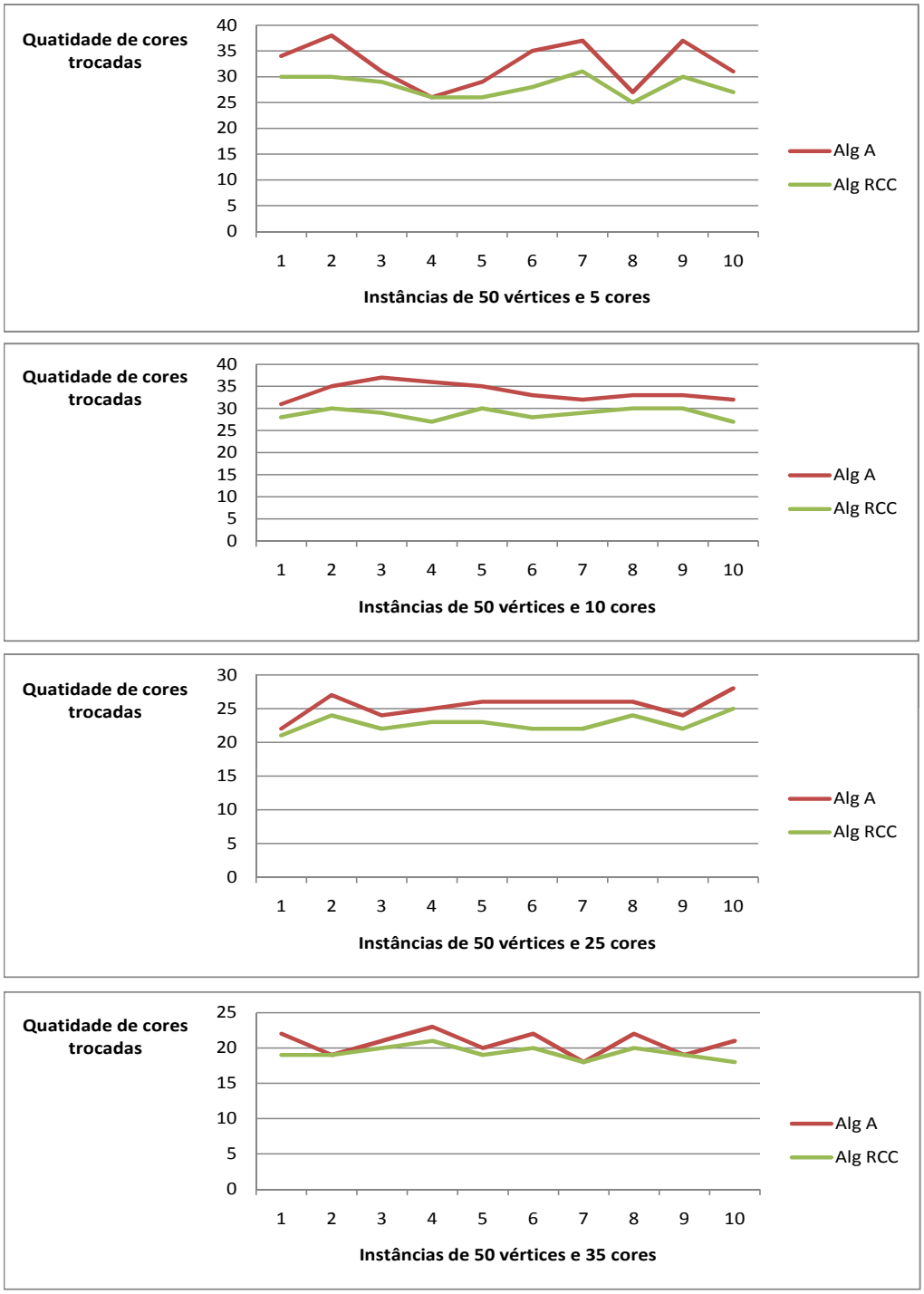

Figura 6.3: *Soluções do algoritmo de* 2*-aproximação e do AlgoritmoRCC para* $n = 50$.

10 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
100	76	67	55,17	Sim	1,13
100	76	68	86,19	Não	1,12
100	75	65	62,61	Não	1,15
100	74	67	68,15	Sim	1,10
100	76	68	240,33	Sim	1,12
100	74	66	166,52	Sim	1,12
100	73	65	77,22	Sim	1,12
100	71	67	79,27	Não	1,06
100	69	63	88,17	Não	1,10
100	78	68	96,65	Não	1,15
média			**102,03**		**1,12**

20 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
100	77	64	57,51	Não	1,20
100	76	65	48,20	Não	1,17
100	70	64	454,55	Sim	1,09
100	68	63	41,88	Não	1,08
100	74	64	581,53	Sim	1,16
100	76	66	310,29	Sim	1,15
100	72	62	264,14	Sim	1,16
100	69	63	294,17	Sim	1,10
100	72	64	131,97	Sim	1,13
100	73	63	274,28	Sim	1,16
média			**245,85**		**1,14**

50 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
100	52	46	34,26	Sim	1,13
100	54	50	35,15	Sim	1,08
100	56	51	53,70	Sim	1,10
100	53	48	13,81	Não	1,10
100	54	51	81,78	Sim	1,06
100	52	51	29,24	Não	1,02
100	57	51	46,89	Sim	1,12
100	54	48	9,96	Não	1,13
100	54	50	47,60	Sim	1,08
100	59	52	58,83	Sim	1,13
média			**41,12**		**1,09**

70 cores					
n	Alg A valor	AlgoritmoRCC valor	AlgoritmoRCC tempo(m)	Branch-and-Bound?	A/RCC
100	48	44	10,55	Não	1,09
100	41	40	108,51	Sim	1,03
100	43	41	12,14	Não	1,05
100	40	38	11,86	Não	1,05
100	43	41	13,26	Não	1,05
100	44	41	13,57	Não	1,07
100	42	41	16,99	Não	1,02
100	43	42	58,92	Sim	1,02
100	44	42	15,64	Não	1,05
100	45	44	11,11	Não	1,02
média			**27,25**		**1,05**

Tabela 6.5: *Algoritmo de* 2*-aproximação* $\times$ *AlgoritmoRCC para* $n = 100$.

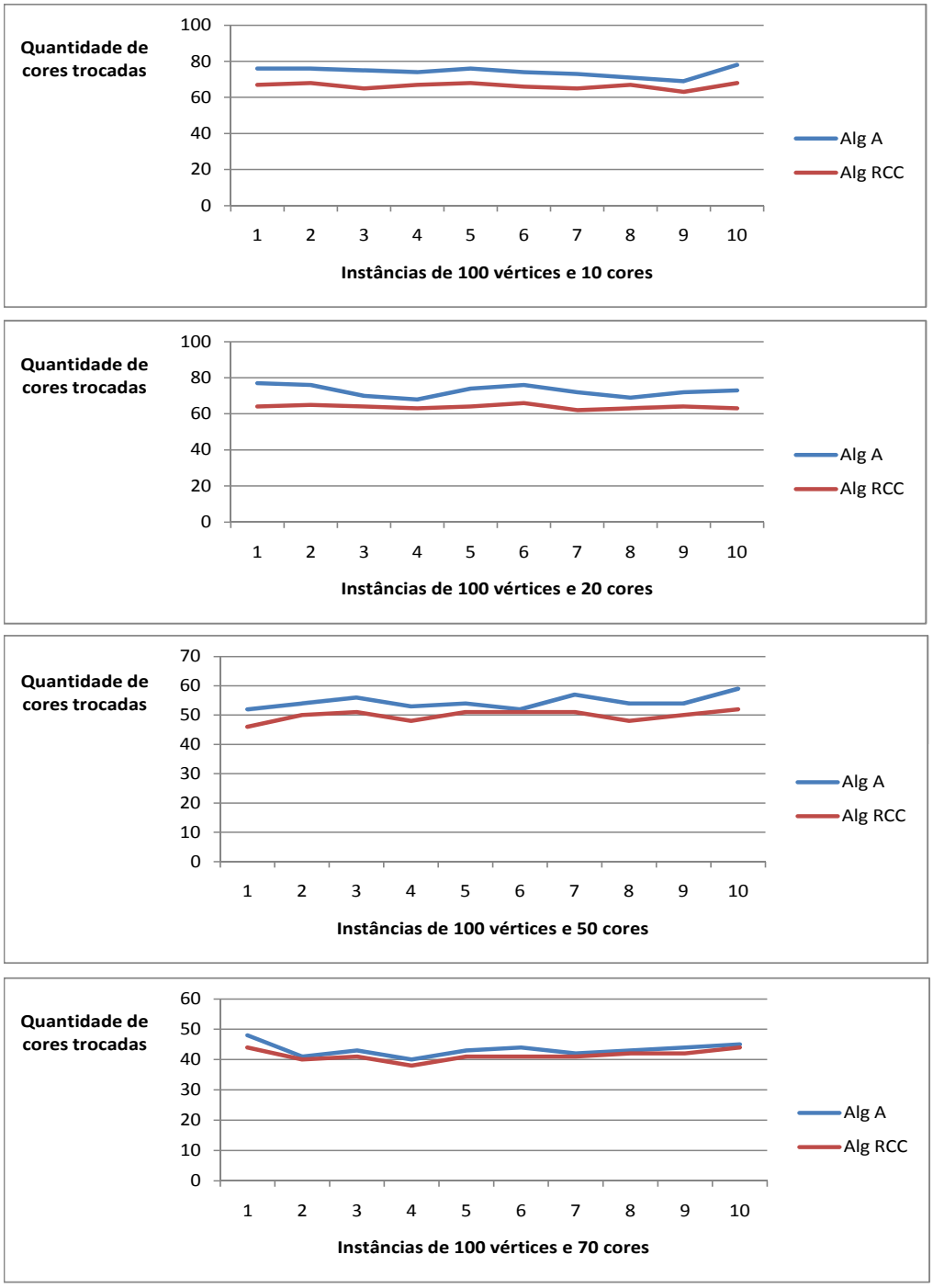

Figura 6.4: *Soluções do algoritmo de* 2*-aproximação e do AlgoritmoRCC para* $n = 100$.

6.3.6 O AlgoritmoRCC modificado e resultados computacionais

Com o objetivo de melhorar o desempenho do AlgoritmoRCC, modificamos tal algoritmo em dois aspectos:

1. Para cada instância decidimos reduzir a quantidade de restrições (que garantem a convexidade) que são adicionadas ao problema inicial a ser otimizado. É claro que a solução inteira devolvida pode não ser convexa e, neste caso, precisamos adicionar novas restrições e reotimizar o problema. Essa estratégia faz com que as restrições que não foram consideradas inicialmente sejam adicionadas posteriormente, apenas se forem necessárias.

2. Como mencionamos na seção anterior, no método *branch-and-bound*, os problemas podem ser resolvidos obedecendo uma ordem que podemos fixar. Basicamente, esta ordem refere-se ao modo de se percorrer os nós da árvore de *branch-and-bound* : em largura ou em profundidade. Aqui, escolhemos percorrer estes nós em profundidade.

Denominamos o AlgoritmoRCC modificado de **AlgoritmoRCC**_*m*.

A seguir, apresentamos nas tabelas 6.6 e 6.7 os resultados computacionais obtidos, e mencionamos algumas conclusões a respeito.

- Após as modificações do AlgoritmoRCC, executamos o AlgoritmoRCC_*m* sobre as mesmas instâncias das tabelas 6.4 e 6.5.
- As tabelas 6.6 e 6.7 mostram os tempos gastos pelos dois algoritmos.
- A quarta e sexta colunas indicam se a instância demandou o uso do método *branch-and-bound*.

▹ Observamos que o AlgoritmoRCC_*m* teve um desempenho bem melhor que o do AlgoritmoRCC.

▹ Notamos pela quarta e sexta colunas que (indicam se a fase *branch-and-bound* foi usada ou não), para a maioria das instâncias, o comportamento dos dois algoritmos relativamente a este aspecto foi semelhante. Isso leva-nos a concluir que, para as instâncias que não entraram no *branch-and-bound*, o que influenciou a melhora do tempo gasto no AlgoritmoRCC_*m* foi a redução da quantidade de restrições que são consideradas no problema inicial (o ponto que diferencia os dois algoritmos).

▹ Por outro lado, verificamos também que, se fixamos a quantidade de restrições consideradas no problema inicial em ambos os algoritmos, nas instâncias que entraram na fase *branch-and-bound* o AlgoritmoRCC_*m* obteve um melhor desempenho que o AlgoritmoRCC. Ou seja, percorrer a árvore de *branch-and-bound* em profundidade, na resolução do problema, leva-nos a encontrar uma solução inteira mais cedo, que pode ser a solução ótima. Essa conclusão não está aparente nas tabelas 6.6 e 6.7, porém realizamos tais testes para optarmos por uma das técnicas que mencionamos no item 2.

Na próxima seção apresentamos os resultados do AlgoritmoRCC_*m* associado a um poliedro que é uma relaxação do poliedro $\mathcal{P}_{nk}$.

6.3.7 Resultados computacionais obtidos com uma relaxação de $\mathcal{P}_{nk}$

Considere o seguinte PL0/1, que é idêntico ao PL0/1 que obtivemos para o problema RCC, descrito no Capítulo 5, a menos do conjunto de restrições (6.3) que é uma relaxação da condição de igualdade.

$$\text{maximize} \sum_{i \in V} \sum_{c \in \mathcal{C}} w_{ic}\, x_{ic}$$

$$\begin{array}{llll} \text{sujeito a} & \sum_{c \in \mathcal{C}} x_{ic} & \leq 1 & i \in V \qquad (6.3) \\ & x_{pc} - x_{qc} + x_{rc} & \leq 1 & 1 \leq p < q < r \leq n,\ c \in \mathcal{C} \qquad (6.4) \\ & x_{ic} & \in \{0,1\} & i \in V,\ c \in \mathcal{C}. \qquad (6.5) \end{array}$$

Considere agora o poliedro

$$\mathcal{P}_{nk_{\leq}} = \text{conv}\{x \in \mathbb{R}^{nk} : x \text{ satisfaz } 6.3, 6.4, 6.5\}.$$

Claramente, o PL0/1 acima é equivalente ao problema de maximizar $\sum_{i \in V} \sum_{c \in \mathcal{C}} w_{ic}\, x_{ic}$ sobre o poliedro $\mathcal{P}_{nk_{\leq}}$.

I- Note que uma solução ótima do PL0/1 acima não é necessariamente o vetor de incidência de uma recoloração convexa de um caminho. De fato, as inequações (6.3) da formulação acima permitem que vértices no caminho fiquem sem receber nenhuma cor, ou seja, a matriz que representa o vetor de incidência desta coloração, pode ter linhas com todas as entradas iguais a zero.

II- Não é difícil **converter** uma solução ótima relativa ao poliedro $\mathcal{P}_{nk_{\leq}}$ em uma solução ótima relativa ao poliedro $\mathcal{P}_{nk}$. Para fazer isto, considera-se a solução 0/1 no

poliedro $\mathcal{P}_{nk_{\leq}}$ e atribui-se a cada vértice sem cor (isto é cada vértice i tal que $x_{i,c} = 0$ para toda cor c) uma cor de modo a garantir que a propriedade de a coloração ser convexa seja mantida/obtida).

Informalmente, a ideia consiste em percorrer o caminho dado (a partir do início), e para cada vértice sem cor (que recebeu a cor 0 na coloração parcial dada pela solução 0/1) repetir a cor do vértice anterior. Ilustramos esse procedimento na Figura 6.5, onde estão representadas as matrizes Γ_a e Γ_b, a primeira delas correspondente a uma solução 0/1 sobre o poliedro $\mathcal{P}_{nk_{\leq}}$, e a segunda correspondente à solução 0/1 sobre o poliedro $\mathcal{P}_{nk}$ (coloração convexa).

Note que as linhas 3, 7 e 9, na matriz Γ_a possuem todas as entradas iguais a zero, ou seja, o vértice 3, 7 e 9 não receberam nenhuma cor. Já na matriz Γ_b, os vértices 3, 7 e 9 receberam a cor do respectivo vizinho anterior.

III- Observe também pela função objetivo do PL0/1 (maximiza a quantidade de cores mantidas) que, em uma solução ótima, se um vértice não recebe nenhuma cor, isto também é contado como troca. Assim, após converter uma solução sobre o poliedro $\mathcal{P}_{nk_{\leq}}$ em uma solução sobre o poliedro $\mathcal{P}_{nk}$, como descrevemos, o valor da solução ótima é o mesmo e a solução obtida após esta conversão também é ótima. Se pudéssemos encontrar uma solução com melhor custo no poliedro $\mathcal{P}_{nk}$ isso iria contradizer a otimalidade da solução obtida no poliedro $\mathcal{P}_{nk_{\leq}}$, já que, $\mathcal{P}_{nk} \subset \mathcal{P}_{nk_{\leq}}$.

$$\Gamma_a = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \qquad \Gamma_b = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Figura 6.5: *Γ_a é um ponto de $\mathcal{P}_{nk_{\leq}}$, enquanto Γ_b é um ponto de $\mathcal{P}_{nk_{\leq}}$ e de $\mathcal{P}_{nk}$.*

Denotamos por AlgoritmoRCC_m' o AlgoritmoRCC_m aplicado ao poliedro relaxado $\mathcal{P}_{nk_{\leq}}$. Além disso, no AlgoritmoRCC_m' adicionamos a rotina de tempo linear, que transforma uma solução 0/1 da relaxação, em uma solução 0/1 para o problema RCC. Apresentamos nas tabelas 6.8 e 6.9, os resultados computacionais obtidos com o AlgoritmoRCC_m' e o algoritmo de 2-aproximação (Alg A).

Implementamos também o algoritmo de $\frac{3}{2}$-aproximação (algoritmo Aprox2RCC apresentado no Capítulo 4), que nomeamos na tabela 6.10 de **Alg B**. Comparamos tal algoritmo de aproximação com o algoritmo AlgoritmoRCC_m' para instâncias do problema 2-RCC. Na tabela 6.10, nomeamos tal algoritmo restrito a instâncias do 2-PRCC por Algoritmo2RCC_m'.

A seguir apresentamos algumas conclusões sobre as tabelas 6.8, 6.9 e 6.10.

- Executamos o AlgoritmoRCC_m' com instâncias de 150 e 200 vértices. O número de cores foi tomado considerando uma porcentagem aproximada de 10%, 20%, 50% e 70% sobre o número de vértices. Nota-se que, à medida que o número de cores aumenta (fixado o número de vértices), o valor devolvido pelo algoritmo de 2-aproximação fornece um limitante inferior cada vez melhor. Isto possivelmente ajuda a diminuir o tempo gasto pelo algoritmo AlgoritmoRCC_m'.

- Observamos que a quantidade de restrições que são adicionadas ao problema inicial é um fator decisivo para se obter solução ótima mais rápido. Nossa escolha se baseou no seguinte procedimento: começamos incluindo pouquíssimas restrições a ponto de obtermos uma solução que não representa uma coloração convexa, e a partir disso, aos poucos, adicionamos mais restrições e reotimizamos até obter uma solução ótima como desejado.

- Com relação ao desempenho do AlgoritmoRCC_m', obtivemos solução ótima para instâncias de 150 vértices, no pior caso, em 105 minutos e no melhor caso em 1 minuto. Já para instâncias de 200 vértices, o tempo no pior caso foi 120 minutos e no melhor caso também foi de aproximadamente 1 minuto.

- Notamos também que o algoritmo de $\frac{3}{2}$-aproximação que desenvolvemos encontrou soluções que não divergiram mais do que 7% da solução ótima, para instâncias de 100 e 200 vértices. Além disso, em média, o Algoritmo2RCC_m' apresentou solução ótima para instâncias de 100 vértices em 3 minutos e para instâncias de 200 vértices em 32 minutos, sendo que no pior caso não ultrapassou 120 minutos.

Comparando os desempenhos dos algoritmos *branch-and-bound* que testamos para o problema RCC, concluímos que os resultados computacionais mais satisfatórios foram obtidos com a estratégia de busca em profundidade, uso de poucas restrições no problema inicial, e uso do poliedro relaxado $\mathcal{P}_{nk_{\leq}}$ (em vez do poliedro exato). Também creditamos ao bom desempenho do algoritmo de separação o fato de o algoritmo *branch-and-bound* ter um desempenho bastante satisfatório.

5 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
50	30	0,06	Sim	0,21	Sim
50	30	0,04	Não	0,11	Não
50	29	0,02	Não	0,08	Não
50	26	0,02	Não	0,08	Não
50	26	0,02	Não	0,07	Não
50	28	1.90	Sim	0,12	Sim
50	31	0,06	Sim	0,12	Não
50	25	0,02	Não	0,04	Não
50	30	0,04	Não	0,13	Não
50	27	0,02	Não	0,06	Não
média		**0,03**		**0,10**	

10 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
50	28	0,05	Não	0,60	Não
50	30	0,21	Sim	0,74	Sim
50	29	0,05	Não	0,21	Não
50	27	0,05	Não	0,14	Não
50	30	0,25	Sim	2,25	Sim
50	28	0,04	Não	0,16	Não
50	29	0,06	Não	0,25	Não
50	30	0,07	Não	0,26	Não
50	30	0,34	Sim	1,02	Sim
50	27	0,05	Não	0,14	Não
média		**0,12**		**0,58**	

25 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
50	21	0,07	Não	0,87	Não
50	24	0,23	Sim	1,25	Não
50	22	0,13	Não	1,55	Não
50	23	0,75	Sim	3,67	Sim
50	23	0,20	Não	1,10	Não
50	22	0,21	Não	0,37	Sim
50	22	0,12	Não	0,19	Sim
50	24	0,09	Não	0,88	Sim
50	22	0,10	Não	0,88	Não
50	25	0,25	Não	1,60	Não
média		**0,22**		**1,24**	

35 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
50	19	0,74	Sim	4,38	Sim
50	19	1,06	Sim	3,32	Sim
50	20	0,12	Não	1,87	Não
50	21	0,40	Não	2,29	Não
50	19	0,22	Não	1,44	Sim
50	20	0,11	Não	4,67	Não
50	18	0,33	Não	0,87	Não
50	20	0,39	Sim	3,05	Sim
50	19	1,50	Sim	0,66	Não
50	18	0,39	Não	0,89	Não
média		**0,53**		**2,34**	

Tabela 6.6: *AlgoritmoRCC* $\times$ *AlgoritmoRCC_m para* $n = 50$.

10 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
100	67	1,48	Sim	55,17	Sim
100	68	1,38	Não	86,19	Não
100	65	1,29	Não	62,61	Não
100	67	1,50	Sim	68,15	Sim
100	68	2,13	Sim	240,33	Sim
100	66	1,51	Sim	166,52	Sim
100	65	0,95	Sim	77,22	Sim
100	67	0,82	Não	79,27	Não
100	63	0,76	Não	88,17	Não
100	68	1,05	Não	96,65	Não
média		**1,29**		**102,03**	

20 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
100	64	1,75	Não	57,51	Não
100	65	1,49	Não	48,20	Não
100	64	2,62	Sim	454,55	Sim
100	63	0,30	Não	41,88	Não
100	64	3,24	Sim	581,53	Sim
100	66	11,97	Sim	310,29	Sim
100	62	3,02	Sim	264,14	Sim
100	63	6,07	Sim	294,17	Sim
100	64	6,76	Sim	131,97	Sim
100	63	5,32	Sim	274,28	Sim
média		**4,25**		**245,85**	

50 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
100	46	0,53	Sim	34,26	Sim
100	50	9,12	Sim	35,15	Sim
100	51	4,94	Sim	53,70	Sim
100	48	1,76	Não	13,81	Não
100	51	4,44	Sim	81,78	Sim
100	51	9,13	Não	29,24	Não
100	51	1,32	Sim	46,89	Sim
100	48	1,49	Não	9,96	Não
100	50	8,13	Sim	47,60	Sim
100	52	1,12	Sim	58,83	Sim
média		**4,20**		**41,12**	

70 cores					
n	**valor**	**AlgoritmoRCC_m tempo(m)**	**Branch-and-Bound?**	**AlgoritmoRCC tempo(m)**	**Branch-and-Bound?**
100	44	1,72	Não	10,55	Não
100	40	1,00	Sim	108,51	Sim
100	41	1,05	Não	12,14	Não
100	38	1,82	Não	11,86	Não
100	41	1,91	Não	13,26	Não
100	41	1,46	Não	13,57	Não
100	41	1,54	Não	16,99	Não
100	42	1,93	Sim	58,92	Sim
100	42	0,81	Não	15,64	Não
100	44	0,91	Não	11,11	Não
média		**1,42**		**27,25**	

Tabela 6.7: *AlgoritmoRCC* $\times$ *AlgoritmoRCC_m para* $n = 100$.

15 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	122	107	5,06	Não	1,14
150	114	106	9,70	Sim	1,08
150	127	105	1,01	Não	1,21
150	125	108	9,00	Sim	1,16
150	115	105	28,66	Sim	1,10
150	118	108	25,14	Não	1,09
150	119	105	31,15	Não	1,13
150	120	109	104,16	Não	1,10
150	122	104	7,77	Não	1,17
150	120	108	29,37	Sim	1,11
média			**25,10**		**1,13**

30 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	110	100	19,26	Não	1,10
150	117	100	12,97	Sim	1,17
150	112	99	14,38	Sim	1,13
150	116	105	56,35	Sim	1,10
150	111	96	11,73	Não	1,16
150	115	101	33,71	Não	1,14
150	111	100	49,49	Sim	1,11
150	113	99	55,49	Sim	1,14
150	115	98	19,07	Não	1,17
150	114	103	45,95	Sim	1,11
média			**31,84**		**1,13**

75 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	77	72	3,75	Não	1,07
150	85	83	16,06	Não	1,02
150	78	74	3,13	Não	1,05
150	89	84	11,40	Sim	1,06
150	84	76	3,67	Não	1,11
150	78	74	3,66	Não	1,05
150	80	75	2,67	Não	1,07
150	85	79	5,24	Não	1,08
150	82	75	4,58	Não	1,09
150	82	78	6,22	Não	1,05
média			**6,04**		**1,07**

105 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	71	66	3,61	Não	1,08
150	66	64	1,87	Não	1,03
150	69	65	2,29	Não	1,06
150	67	64	2,79	Não	1,05
150	66	64	2,19	Não	1,03
150	67	66	10,88	Não	1,02
150	77	71	3,11	Não	1,08
150	71	66	6,33	Não	1,08
150	68	64	5,06	Não	1,06
150	65	61	4,71	Não	1,07
média			**4,28**		**1,06**

Tabela 6.8: *Algoritmo de* 2*-aproximação* $\times$ *AlgoritmoRCC_m′ para* $n = 150$.

15 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	122	107	5,06	Não	1,14
150	114	106	9,70	Sim	1,08
150	127	105	1,01	Não	1,21
150	125	108	9,00	Sim	1,16
150	115	105	28,66	Sim	1,10
150	118	108	25,14	Não	1,09
150	119	105	31,15	Não	1,13
150	120	109	104,16	Não	1,10
150	122	104	7,77	Não	1,17
150	120	108	29,37	Sim	1,11
média			**25,10**		**1,13**

30 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	110	100	19,26	Não	1,10
150	117	100	12,97	Sim	1,17
150	112	99	14,38	Sim	1,13
150	116	105	56,35	Sim	1,10
150	111	96	11,73	Não	1,16
150	115	101	33,71	Não	1,14
150	111	100	49,49	Sim	1,11
150	113	99	55,49	Sim	1,14
150	115	98	19,07	Não	1,17
150	114	103	45,95	Sim	1,11
média			**31,84**		**1,13**

75 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	77	72	3,75	Não	1,07
150	85	83	16,06	Não	1,02
150	78	74	3,13	Não	1,05
150	89	84	11,40	Sim	1,06
150	84	76	3,67	Não	1,11
150	78	74	3,66	Não	1,05
150	80	75	2,67	Não	1,07
150	85	79	5,24	Não	1,08
150	82	75	4,58	Não	1,09
150	82	78	6,22	Não	1,05
média			**6,04**		**1,07**

105 cores					
n	Alg A valor	AlgoritmoRCC_m' valor	AlgoritmoRCC_m' tempo(m)	Branch-and-Bound?	A/RCC_m'
150	71	66	3,61	Não	1,08
150	66	64	1,87	Não	1,03
150	69	65	2,29	Não	1,06
150	67	64	2,79	Não	1,05
150	66	64	2,19	Não	1,03
150	67	66	10,88	Não	1,02
150	77	71	3,11	Não	1,08
150	71	66	6,33	Não	1,08
150	68	64	5,06	Não	1,06
150	65	61	4,71	Não	1,07
média			**4,28**		**1,06**

Tabela 6.9: *Algoritmo de* 2*-aproximação* $\times$ *AlgoritmoRCC_m′ para* $n = 200$.

50 cores					
n	Alg B	Algoritmo2RCC_m'		Branch-and-Bound?	B/2RCC_m'
	valor	valor	tempo(m)		
100	42	41	1,76	Não	1,02
100	43	41	0,97	Não	1,05
100	45	43	5,23	Sim	1,05
100	45	42	7,47	Não	1,07
100	46	42	0,97	Não	1,10
100	46	44	1,41	Não	1,05
100	46	42	0,82	Não	1,10
100	46	38	2,47	Sim	1,21
100	45	43	1,40	Sim	1,05
100	42	41	1,57	Sim	1,02
média			**2,41**		**1,07**

75 cores					
n	Alg B	Algoritmo2RCC_m'		Branch-and-Bound?	B/2RCC_m'
	valor	valor	tempo(m)		
100	38	37	2,16	Não	1,03
100	35	33	5,04	Não	1,06
100	36	35	1,89	Não	1,03
100	35	35	4,83	Não	1,00
100	34	34	1,65	Não	1,00
100	34	32	1,59	Não	1,06
100	34	33	3,12	Não	1,03
100	37	37	2,36	Não	1,00
100	37	36	1,29	Não	1,03
100	38	36	1,44	Não	1,06
média			**2,54**		**1,03**

100 cores					
n	Alg B	Algoritmo2RCC_m'		Branch-and-Bound?	B/2RCC_m'
	valor	valor	tempo(m)		
200	91	87	2,35	Não	1,05
200	98	92	4,20	Não	1,07
200	94	90	9,41	Não	1,04
200	94	87	4,23	Sim	1,08
200	94	90	27,70	Não	1,04
200	95	89	27,29	Não	1,07
200	93	87	3,36	Não	1,07
200	90	87	117,99	Não	1,03
200	94	87	4,39	Não	1,08
200	95	87	8,58	Não	1,09
média			**20,95**		**1,06**

150 cores					
n	Alg B	Algoritmo2RCC_m'		Branch-and-Bound?	B/2RCC_m'
	valor	valor	tempo(m)		
200	72	71	52,62	Não	1,01
200	78	78	76,27	Não	1,00
200	72	70	17,35	Não	1,03
200	74	74	64,66	Não	1,00
200	71	69	11,62	Não	1,03
200	73	73	43,56	Não	1,00
200	73	72	19,87	Não	1,01
200	73	70	27,98	Não	1,04
200	77	75	73,31	Não	1,03
200	79	76	31,33	Não	1,04
média			**41,86**		**1,02**

Tabela 6.10: *Algoritmo de* $\frac{3}{2}$*-aproximação* $\times$ *Algoritmo2RCC_m' para* $n = 100, 200$.

Capítulo 7

Considerações finais

Neste trabalho, apresentamos algoritmos para o problema de recoloração convexa de caminhos, que é sabidamente um problema NP-difícil. Desenvolvemos um algoritmo de $\frac{3}{2}-$aproximação para o caso especial, que também é NP-difícil, em que as instâncias são caminhos nos quais cada cor ocorre no máximo duas vezes. Trata-se de um algoritmo simples e fácil de ser implementado, porém provar que sua razão de aproximação é no máximo $\frac{3}{2}$ foi uma tarefa relativamente complicada. Alguns estudos que fizemos, e que não mencionamos em nenhuma parte deste trabalho, nos mostraram que talvez seja possível obter um algoritmo com razão de aproximação $\frac{4}{3}$ para este caso restrito. Talvez venhamos a provar isso, mais como um desafio teórico.

Em termos de pesquisa futura, achamos porém que seria um grande desafio investigar a possibilidade de obter algum algoritmo de aproximação para o problema de recoloração convexa de grafos arbitrários.

Exploramos também o problema de recoloração convexa de caminhos através de métodos poliédricos. Desenvolvemos uma formulação inteira para o problema, exibimos classes de inequações válidas para o poliedro associado a esta formulação e mostramos que várias delas definem facetas. Implementamos um algoritmo de programação dinâmica, polinomial, que resolve o problema da separação para estas facetas. Associado a este algoritmo de separação implementamos também um algoritmo *branch-and-cut* para o problema.

Os resultados experimentais mostraram que estas inequações que definem facetas foram fundamentais para encontrarmos solução ótima inteira mais rápido. Além disso, observamos que, quando executamos somente o algoritmo de separação, independente do método *branch-and-cut*, nos casos em que a solução inteira ótima não foi encontrada, este algoritmo nos deixa a menos de $1,5\%$ do valor da solução ótima inteira. Este resultado foi observado comparando este resultado inicial com a solução ótima inteira devolvida após a execução do método *branch-and-cut*.

Um desafio, em termos do estudo do poliedro que definimos, seria encontrar (se

existentes) inequações que definem facetas, que relacionem variáveis de várias cores. Não está muito claro se estas seriam tão importantes quanto as inequações de convexidade-generalizada que obtivemos, mas seria interessante descobrir isso.

Referências Bibliográficas

[1] Richa Agarwala and David Fernandez-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM J. Comput.*, 23:1216–1224, December 1994.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[3] Reuven Bar-Yehuda, Ido Feldman, and Dror Rawitz. Improved approximation algorithm for convex recoloring of trees. In *Approximation and online algorithms*, volume 3879 of *Lect. Notes in Comput. Sci.*, pages 55–68. Springer, Berlin, 2006.

[4] Alexander Bockmayr, Friedrich Eisenbrand, Mark Hartmann, and Andreas S. Schulz. On the chvátal rank of polytopes in the 0/1 cube. *Discrete Applied Mathematics*, 98(1-2):21 – 27, 1999.

[5] Hans L. Bodlaender, Michael R. Fellows, Michael T. Hallett, H. Todd Wareham, and Tandy J. Warnow. The hardness of perfect phylogeny, feasible register assignment and other problems on thin colored graphs. *Theoret. Comput. Sci.*, 244(1-2):167–188, 2000.

[6] Hans L. Bodlaender, Michael R. Fellows, Michael A. Langston, Mark A. Ragan, Frances A. Rosamond, and Mark Weyer. Quadratic kernelization for convex recoloring of trees. In *Proceedings COCOON 2007*, volume 4598 of *Lecture Notes in Comput. Sci.*, pages 86–96. Springer, Berlin, 2007.

[7] J.A. Bondy and U.S.R. Murty. *Graph Theory*. Springer, 2008.

[8] M.H. Carvalho, M.R. Cerioli, R. Dahab, P. Feofiloff, C.G. Fernandes, C.E. Ferreira, F.K. Miyazawa, J.C. de Pina, J. Soares, and Y. Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Publicações Matemàticas do IMPA, 2001.

[9] Benny Chor, Michael Fellows, Mark A. Ragan, Igor Razgon, Frances Rosamond, and Sagi Snir. Connected coloring completion for general graphs: Algorithms and complexity. In *Proceedings COCOON 2007*, volume 4598 of *Lecture Notes in Comput. Sci.*, pages 75–85. Springer, Berlin, 2007.

[10] R.G. Downey and M.R. Fellows. *Parameterized complexity*. Monographs in Computer Science. Springer-Verlag, New York, 1999.

[11] Carlos E. Ferreira and Yoshiko Wakabayashi. *Combinatória Poliédrica e Planos-de-Corte Faciais*. Livro da X Escola de Computação-UNICAMP, 1º edition, 1996.

[12] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[13] Frank Kammer and Torsten Tholey. The complexity of minimum convex coloring. In *Proceedings ISAAC 2008*, volume 5369 of *Lect. Notes in Comput. Sci.*, pages 16–27. Springer, Heidelberg, Berlin, 2008.

[14] Iyad Kanj and Dieter Kratsch. Convex recoloring revisited: Complexity and exact algorithms. In *Algorithms and data structures*, volume 5609 of *Lect. Notes in Comput. Sci.*, pages 388–397. Springer, Berlin, 2009.

[15] Karla R. Lima and Yoshiko Wakabayashi. Convex recoloring of paths. In *The VI Latin-American Algorithms, Graphs, and Optimization Symposium*, volume 37 of *Electron. Notes Discrete Math.*, pages 165–170. Elsevier Sci. B. V., Amsterdam, 2011.

[16] Shlomo Moran and Sagi Snir. Efficient approximation of convex recolorings. In *Approximation, randomization and combinatorial optimization*, volume 3624 of *Lect. Notes in Comput. Sci.*, pages 192–208. Springer, Berlin, 2005.

[17] Shlomo Moran and Sagi Snir. Efficient approximation of convex recolorings. *J. Comput. System Sci.*, 73(7):1078–1089, 2007.

[18] Shlomo Moran and Sagi Snir. Convex recolorings of strings and trees: definitions, hardness results and algorithms. *J. Comput. System Sci.*, 74(5):850–869, 2008.

[19] Shlomo Moran, Sagi Snir, and Sung Wing-Kin. Partial convex recolorings of trees and galled networks: Tight upper and lower bounds. ACM Transactions on Algorithms, to appear.

[20] Oriana Ponta, Falk Hüffner, and Rolf Niedermeier. Speedind up dynamic programming for some np-hard graph recoloring problems. In *Proceedings TAMC*, volume 4978 of *Lecture Notes in Comput. Sci.*, pages 490–501. Springer, Heidelberg, Berlin, 2008.

[21] Igor Razgon. A $2^{O(k)}\text{poly}(n)$ algorithm for the parameterized convex recoloring problem. *Inform. Process. Lett.*, 104(2):53–58, 2007.

[22] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience series in discrete mathematics, 1º edition, 1986.

[23] Laurence A. Wolsey. *Integer Programming*. Wiley-Interscience series in discrete mathematics and optimization, 1º edition, 2008.