UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO JORGE DE ABREU FIGUEREDO

O PROBLEMA DA FLORESTA GERADORA k-ROTULADA

FORTALEZA

2020

PEDRO JORGE DE ABREU FIGUEREDO

O PROBLEMA DA FLORESTA GERADORA *k*-ROTULADA

Dissertação apresentada ao Curso de do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Manoel Bezerra Campêlo Neto

FORTALEZA

2020

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

F484p Figueredo, Pedro Jorge de Abreu.
O Problema da Floresta Geradora k-Rotulada / Pedro Jorge de Abreu Figueredo. – 2020.
99 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2020.
Orientação: Prof. Dr. Manoel Bezerra Campêlo Neto.

1. Floresta Geradora k-Rotulada. 2. Branch&Cut. 3. Backtracking. 4. Programação Inteira. 5. Programação Paralela. I. Título.

CDD 005

PEDRO JORGE DE ABREU FIGUEREDO

O PROBLEMA DA FLORESTA GERADORA k-ROTULADA

Dissertação apresentada ao Curso de do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciência da Computação. Área de Concentração: Ciência da Computação

Aprovada em: 10 de Fevereiro de 2020

BANCA EXAMINADORA

Prof. Dr. Manoel Bezerra Campêlo Neto (Orientador)
Universidade Federal do Ceará (UFC)

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

Prof. Dr. Lucídio dos Anjos Formiga Cabral
Universidade Federal da Paraíba (UFPB)

Prof. Dr. Marcos José Negreiros Gomes
Universidade Estadual do Ceará (UECE)

AGRADECIMENTOS

Ao Prof. Dr. Manoel Bezerra Campêlo Neto por me orientar em minha dissertação de mestrado.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Aos bibliotecários da Universidade Federal do Ceará: Eliene Maria Vieira de Moura, Francisco Edvander Pires Santos, Izabel Lima dos Santos, Juliana Soares Lima, Kalline Yasmin Soares Feitosa pela revisão e discussão da formatação utilizada neste *template*.

Ao aluno Thiago Nascimento do curso de ciência da computação da Universidade Estadual do Ceará que elaborou o *template* do qual este trabalho foi adaptado para Universidade Federal do Ceará.

Aos meus parentes, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

E à Fundação Cearense de Apoio ao Desenvolvimento (Funcap), na pessoa do Presidente Tarcísio Haroldo Cavalcante Pequeno pelo financiamento da pesquisa de mestrado via bolsa de estudos.

RESUMO

Neste trabalho, estudamos o problema da Floresta Geradora k-Rotulada. Dentre os problema em grafos coloridos em arestas, esse tem como objetivo encontrar um subgrafo gerador que é uma floresta com a menor quantidade de componentes e com um fator limitante inteiro positivo k para o número de rótulos distintos usados. O problema é NP-Difícil, por ser generalização do problema da Árvore Geradora Minimamente Rotulada, e a maioria dos métodos propostos pela literatura são baseados em metaheurísticas. Estudamos características do problema e relacionamos com a literatura pertinente em grafos coloridos. A partir das propriedades estudadas, estabelecemos uma classe na qual o problema é resolvido em tempo polinomial. Também exploramos caracterizações do problema para propor três modelos matemáticos de programação inteira mista binária e inteira binária. Propomos métodos *Branch&Cut* para resolver o problema de forma exata, usando um algoritmo para separação de Cortes Coloridos. Além disso, propomos uma paralelização para o único método exato disponível na literatura, que aplica um procedimento *backtracking*. Apresentamos experimentos computacionais preliminares com os modelos e o método *backtracking*, usando instâncias de teste sugeridas pela literatura. Aplicamos melhorias para os métodos de resolução, explorando desigualdades válidas como cortes, regras de *branching*, poda e limites. Além disso, aplicamos decomposição de Benders ao modelo via fluxo proposto. Em seguida, testamos os modelos para grafos com parâmetro de densidade de arestas diferenciados, porém pertinentes ao problema. Desses testes, mostramos as melhores configurações para os modelos e *Cplex* encontradas.

Palavras-chave: Floresta Geradora k-Rotulada. *Branch&Cut*. *Backtracking*. Programação Inteira. Programação Paralela.

ABSTRACT

In this work, we study the k-Labeled Spanning Forest Problem. Among the problems in edge labeled graphs, this one aims at finding a spanning subgraph that is a forest with the least amount of components and with a positive integer limiting factor k for the number of different labels used. The problem is NP-Hard, as it is a generalization of the Minimal Labeled Spanning Tree Problem, and most of the methods proposed in the literature are based on metaheuristics. We study the characteristics of the problem and relate it to the relevant literature in labeled graphs. Through the studied properties, we establish a class on which the problem is solved in polynomial time. We also explore the characterizations of the problem to propose three mathematical models in 0-1 integer and mixed-integer programming. We propose Branch&Cut methods to solve the problem exactly, using an algorithm to separate Colorful Cuts. Moreover, we propose a parallelization for the only exact method available in the literature, which applies a backtracking procedure. We present preliminary computational experiments with the models and the backtracking method, using test instances suggested by the literature. We apply improvements to the resolution methods by exploring valid inequalities as cuts, branching rules, pruning, and bounds. Besides, we apply Benders decomposition to the proposed flow model. Then, we test the models on graphs with different edge density parameters, still pertinent to the problem. Among these tests, we show the best-found configurations for the models and Cplex.

Keywords: k-Labeled Spanning Forest. Branch&Cut. Backtracking. Integer Programming. Parallel Programming.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

DFS	*Depth First Search*
FGkR	Floresta Geradora k-Rotulada
GCA	Grafo colorido em Arestas
MLSTP	*Minimum Labeling Spanning Tree Problem*
MVCA	*Maximum Vertex Covering Algorithm*
PL	Programa Linear
PLB	Programa Linear Binário
PLI	Programa Linear Inteiro Misto
PLI	Programa Linear Inteiro
TBB	*Threading Building Blocks*
VNS	*Variable Neighborhood Search*

SUMÁRIO

1 INTRODUÇÃO

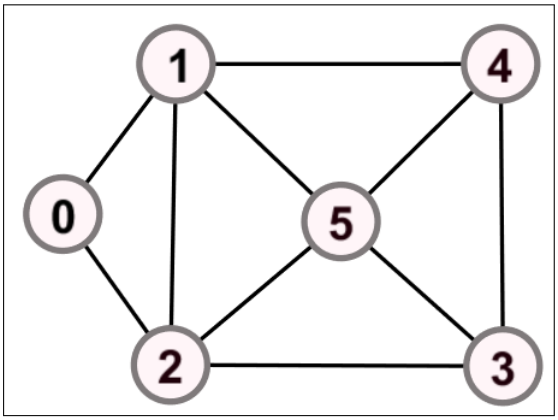
Neste capítulo, serão vistos conceitos básicos de grafos, relacionados ao problema principal, bem como a definição deste de acordo com a literatura. Serão recapitulados aspectos do problema, como complexidade assintótica, uma relaxação e características gerais de métodos de solução já propostos. Além disso, serão exploradas as motivações, os objetivos que se deseja alcançar com este trabalho e, por fim, será colocado como este está organizado.

1.1 Conceitos básicos

Um grafo G pode ser matematicamente definido por uma tripla $G = (V, E, f)$, onde V é conjunto de vértices, E é o conjunto de arestas e $f : E \to V \times V$ é uma função que descreve cada aresta como um par não ordenado de vértices. Os conjuntos de vértices e arestas de um grafo G e sua função de mapeamento serão também referenciados por $V(G)$, $E(G)$ e f_G, respectivamente. Para diferenciar do caso em que a imagem de f são pares ordenados de V, esse tipo de grafo é chamado de não direcionado. Essa estrutura pode ser utilizada para modelar diversos problemas de otimização como caixeiro viajante, emparelhamento máximo, coloração de vértices e arestas etc.

Ao desenhar um grafo, representam-se os vértices como pontos e arestas como linhas conectando o par de vértices associados, conforme pode ser visto na Figura 1. O grafo é dito simples quando a função f é injetora (não possui arestas múltiplas) e sua imagem não contém pares de vértices iguais (não possui laços). Quando f não é injetora, costuma-se chamar o grafo de multigrafo.

Figura 1 – Grafo simples e não direcionado



Fonte: elaborado pelo autor (2020).

Por simplicidade, cada vértice de $V(G)$ será identificado por um número inteiro positivo único, ou seja, $V = \{1,2,3,4,...,|V(G)|\}$. E cada aresta de $E = \{e_1,\ldots,e_{|E(G)|}\}$ será representada diretamente pelos vértices associados, ou seja, da forma $e_i = (v,u)$, com $v,u \in V(G)$, onde, na verdade, $(u,v) = f(e_i)$. Assim, para fins práticos, omite-se f, escrevendo $G = (V,E)$. Se $e = (v,u) \in E(G)$, significa que u e v são vizinhos (ou adjacentes) e são as extremidades de e. Assim, no caso de grafos simples E é um conjunto, e para multigrafo E é um multiconjunto.

Antes de descrever o problema principal, são definidas algumas noções básicas sobre um grafo, que serão referenciadas. Um grafo H é dito subgrafo de G ($H \subseteq G$), se $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ e $f_H : E(H) \to V(H) \times V(H)$ coincide com $f_G : E(G) \to V(G) \times V(G)$. Para $H \subseteq G$, H é gerador se $V(H) = V(G)$ e H é induzido se $E(H) = \{(u,v) \in E(G) : u,v \in V(H)\}$.

Um passeio em um grafo simples $G = (V,E)$ é uma sequência alternada de vértices e arestas da forma $P =< v_0e_1v_1e_2...e_jv_j >$ tal que $e_i = (v_{i-1},v_i)$. Diz-se que P é um passeio entre v_0 e v_j. Um caminho é um passeio onde não há repetição de vértices. E quando em um passeio $C =< v_0e_1v_1e_2...e_jv_j >$ não há repetição de vértice, a não ser $v_0 = v_j$, C é chamado de ciclo.

Um par de vértices é dito conectado se existe um caminho entre eles. Além disso, se todo par de vértices é conectado, o grafo é dito conexo; caso contrário, ele é desconexo. As componentes conexas de um grafo consistem em subgrafos disjuntos conexos maximais (em vértices e arestas). Uma árvore é um grafo minimalmente (em arestas) conexo, e uma árvore geradora de um grafo consiste em um subgrafo gerador que é uma árvore. Vale ressaltar que se pode definir de forma equivalente uma árvore como um grafo conexo e que não possui ciclos. O número de componentes conexas de um grafo G será denotado por $\mathscr{W}(G)$. Seguindo a definição de (BONDY, 2010), uma floresta é um grafo que não possui ciclos, e uma floresta geradora é um subgrafo gerador que é uma floresta. Note que, se H é floresta geradora de G, então $\mathscr{W}(H) \geq \mathscr{W}(G)$, sendo a igualdade sempre satisfeita quando H é maximal em arestas.

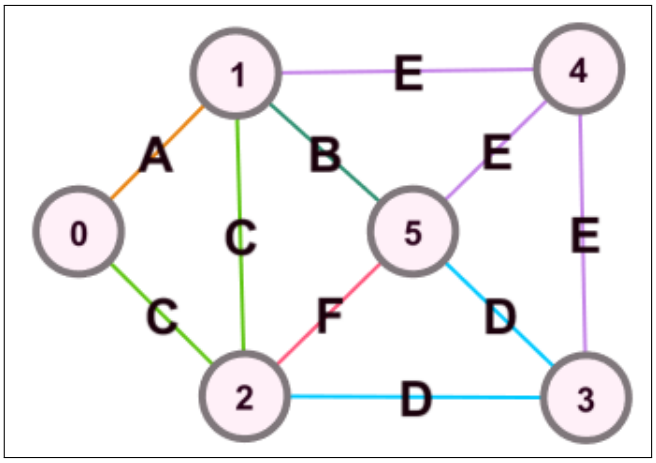
Um grafo é dito ponderado em arestas (ou vértices) quando valores numéricos são atribuídos a esses elementos, para representar peso, custo, capacidade e outros. Diferente de grafos ponderados, grafos coloridos têm cores (rótulos) associadas a seus elementos. No caso de arestas, essas cores mensuram agora qualitativamente as relações entre os vértices extremidade e marcam diferenças entre elas.

Uma quantidade considerável de trabalhos foi e ainda é dedicada a problemas em grafos coloridos, segundo (GRANATA *et al.*, 2013). Usualmente, problemas em grafos coloridos procuram encontrar subestruturas homogêneas ou heterogêneas segundo esses rótulos. Eles são

importantes para campos de pesquisa em redes de comunicação, redes de transporte multimodal, dentre outros.

Um grafo simples e não direcionado é dito colorido em arestas (GCA) quando é da forma $G = (V, E, L, l)$ onde V é o conjunto de vértices, E o conjunto de arestas, L um conjunto de rótulos e $l : E \rightarrow L$ é uma função de coloração das arestas de G, que associa um rótulo para cada aresta de G. Por padrão, $l(e)$ é representado por uma letra maiúscula, ou seja, $L = \{A, B, C, D, ...\}$. Ver ilustração na Figura 2.

Figura 2 – Grafo Colorido em Arestas



Fonte: elaborado pelo autor (2020).

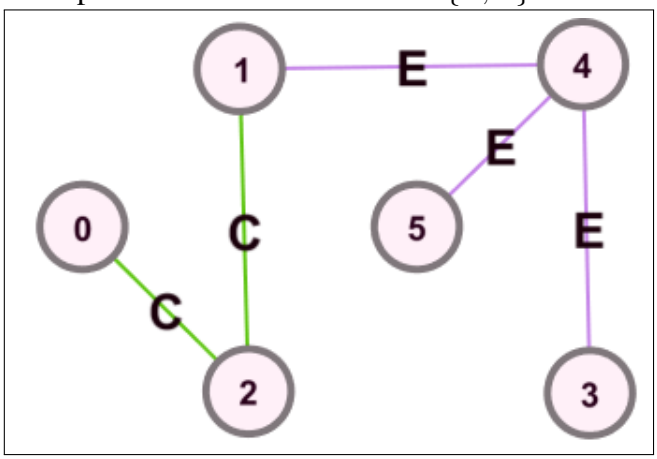
1.2 O Problema

Dado um GCA e um inteiro positivo k, o problema da FGkR consiste em encontrar uma floresta geradora com a menor quantidade de árvores (componentes) e número de cores (rótulos) distintas menor ou igual a k (CERULLI *et al.*, 2014).

Como exemplo, considere o problema FGkR definido sobre o GCA da Figura 2 e $k = 2$. A solução ótima possui uma única componente, como apresentada na Figura 3. Por outro lado, ao considerar o mesmo GCA mas $k = 1$, a solução ótima passa a ter duas componentes, estando representada pela Figura 4. Veja que, ao excluir a aresta (1,4) na Figura 2, diferentes e múltiplas soluções são encontradas.

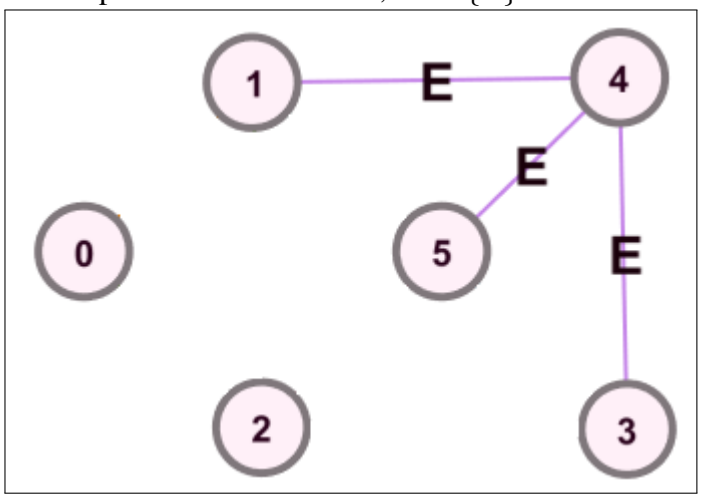
A prova de complexidade assintótica e os métodos propostos até agora para resolver o FGkR estão diretamente relacionados ao problema da Árvore Geradora Minimamente Rotulada (*Minimum Labeling Spanning Tree Problem* (MLSTP), em inglês), cujo objetivo é encontrar uma árvore geradora com a menor quantidade de cores de um GCA. Em (CHANG; SHING-

Figura 3 – Solução ótima para FGkR com $k = 2$. $L^* = \{C, E\}$.



Fonte: elaborado pelo autor (2020).

Figura 4 – Solução ótima para FGkR com $k = 1$, $L^* = \{E\}$.



Fonte: elaborado pelo autor (2020).

JIUAN, 1997) provou-se que a versão de decisão do MLSTP é um problema NP-Completo, e esse resultado pode ser estendido para o FGkR, através de uma redução direta apresentada por (CERULLI *et al.*, 2014). Esses autores consideram os seguintes problemas de decisão associados a MLSTP e FGkR, respectivamente:

- Dado um $k \leq |L|$, existe um subgrafo gerador conexo H de G contendo arestas com no máximo k rótulos distintos?
- Dados $k \leq |L|$ e $w \leq |V|$, existe um subgrafo gerador H de G contendo arestas com no máximo k rótulos diferentes e tal que $\mathscr{W}(H) \leq w$?

Os autores observam que a versão de decisão do MLSTP é um caso especial daquela de FGkR para $w = 1$. Logo, como o MLSTP (decisão) é NP-Completo, então o FGkR (decisão) também é NP-Completo. Vale ressaltar que o problema de decisão do FGkR não exige que a solução seja uma árvore, esse fato é explicado na Proposição 1.2.2 da subseção 1.2.1.

(CERULLI *et al.*, 2014), (CONSOLI; PéREZ, 2015) e (CONSOLI *et al.*, 2017) apresentam trabalhos com enfoque em métodos meta-heurísticos para resolver o FGkR, adaptando algoritmos similares desenvolvidos para o MLSTP. Dentre as meta-heurísticas utilizadas nesses três trabalhos encontram-se as bem conhecidas Arrefecimento Simulado, Algoritmo Genético, GRASP e Busca em Vizinhança Variável (*Variable Neighborhood Search* (VNS), em inglês). Além delas, foram também avaliadas nesses trabalhos as variantes do VNS chamadas Intel-VNS e Co-VNS (propostas em (CONSOLI *et al.*, 2013) para o MLSTP) e Método Piloto (ver descrição dessa meta-heurística em (VOBS *et al.*, 2005)).

Com respeito a métodos exatos, ao nosso conhecimento, o único trabalho existente na literatura também se deve a (CERULLI *et al.*, 2014), que propõem um algoritmo enumerativo, usando *backtracking*.

1.2.1 Formalização

Esta subseção formaliza definições e notações básicas associadas ao problema.

Seja $G = (V, E, L, l)$ um GCA. Dado $E' \subseteq E$, denota-se por $l(E')$ o subconjunto de rótulos das arestas em E', ou seja, $l(E') = \{l(e) \mid e \in E'\}$. Assim, ao se referir a um subgrafo $H \subseteq G$, com conjunto de vértices $V' \subseteq V$ e conjunto de arestas $E' \subseteq E \cap (V' \times V')$, está referindo-se ao GCA $H = (V', E', l(E'), l')$, onde l' é a função l restrita a E'. Por simplicidade, vamos usar l em lugar de l' na descrição de H, ficando entendido que a função se restringe às arestas do subgrafo.

Por outro lado, dado um subconjunto de rótulos $L' \subseteq L$, denota-se por $G[L']$ o subgrafo gerador de G induzido pelos rótulos em L', ou melhor, pelas arestas $E(L') = \{e \in E | l(e) \in L'\}$. Em outros termos, $G[L'] = (V, E(L'), L', l)$.

Seja $\mathscr{F}_G(k)$ o conjunto de todas as florestas geradoras de G com no máximo k rótulos distintos, ou seja, $\mathscr{F}_G(k)$ é o conjunto de todas as soluções válidas para o problema. Assim, define-se formalmente o problema, da seguinte forma:

Definição 1.2.1. Seja $G = (V, E, L, l)$ um GCA e $k \in \mathbb{Z}^+$. O problema FGkR consiste em encontrar uma floresta geradora $F^* \in \mathscr{F}_G(k)$ tal que $\mathscr{W}(F^*) \leq \mathscr{W}(F)$, $\forall F \in \mathscr{F}_G(k)$.

Veja que o problema pode ser relaxado, sem perda de otimalidade, como determinar um subgrafo gerador induzido por até k cores, com a menor quantidade de componentes. Sendo assim, pode-se redefinir o problema como a seguir (CERULLI *et al.*, 2014).

Proposição 1.2.2. *Seja $G = (V,E,L,l)$ um GCA e $k \in \mathbb{Z}^+$. O FGkR equivale a encontrar um subconjunto de rótulos $L^* \subseteq L$ tal que $|L^*| \leq k$ e $\mathscr{W}(G[L^*]) \leq \mathscr{W}(G[L'])$, $\forall L' \subseteq L, |L'| \leq k$.*

Demonstração. Primeiro, seja $F^* \in \mathscr{F}_G(k)$ tal que $\mathscr{W}(F^*) \leq \mathscr{W}(F)$, para todo $F \in \mathscr{F}_G(k)$ (isso é, uma solução ótima para o FGkR). Seja $L^* = l(E(F^*))$ o conjunto de rótulos distintos de F^*. Mostra-se que L^* é solução do problema equivalente. Note que $|L^*| \leq k$, pois F^* tem no máximo k rótulos. Além disso, F^* é floresta geradora de $G[L^*]$, pois só contém arestas de $G[L^*]$, e tem o menor número de componentes, dado que esta é ótima. Logo, $\mathscr{W}(F^*) = \mathscr{W}(G[L^*])$. Tome $L' \subseteq L$ com $|L'| \leq k$ e F' uma floresta geradora de $G[L']$, maximal em arestas. Como $F' \in \mathscr{F}_G(k)$, segue-se que $\mathscr{W}(G[L^*]) = \mathscr{W}(F^*) \leq \mathscr{W}(F') = \mathscr{W}(G[L'])$. Portanto, L^* satisfaz a proposição.

Reciprocamente, seja L^* satisfazendo a proposição. Seja F^* uma floresta geradora de $G[L^*]$, maximal em arestas, de modo que $\mathscr{W}(F^*) = \mathscr{W}(G[L^*])$. Mostra-se que F^* é solução ótima para FGkR. Note que $F^* \in \mathscr{F}_G(k)$, pois $|L^*| \leq k$. Tome $F' \in \mathscr{F}_G(k)$ e $L' = l(E(F'))$. Como $|L'| \leq k$ e F' é floreta geradora de $G[L']$, segue-se que $\mathscr{W}(F^*) = \mathscr{W}(G[L^*]) \leq \mathscr{W}(G[L']) \leq \mathscr{W}(F')$. Logo, F^* satisfaz a Definição 1.2.1. □

Veja que, se $k \geq |L|$, o problema torna-se fácil. É suficiente encontrar uma floresta geradora com o mesmo número de componentes de G. Por outro lado, quando $k \leq |L|$, podemos fixar, sem perda de generalidade, o tamanho do conjunto de cores L^* da solução ótima em *exatamente* k, sem alterar a otimalidade. De fato, se $|L^*|$ é menor que k, adicionar cores, e portanto induzir mais arestas de G, não aumenta a cardinalidade de $\mathscr{W}(G[L^*])$, levando a uma solução igualmente ótima.

1.2.2 Motivação

Problemas como o FGkR são relevantes pois ajudam a encontrar estruturas homogêneas ou heterogêneas dentro de estruturas que podem ser representadas por grafos, obedecendo algum fator que limite seu tamanho. Esse fator é essencial para lidar com parâmetros no mundo real que não podem ser violados. Por exemplo, com o surgimento de novas tecnologias de redes sem fio, como redes de celulares e **WiFi**, pessoas são expostas a uma maior quantidade de campos eletromagnéticos (CEMs) em diferentes frequências. A exposição a CEMs e como eles afetam à saúde têm sido um desafio constante de pesquisa para a comunidade científica (WIART *et al.*, 2019).

Através do FGkR é possível modelar esse cenário usando k como limite do número de frequências a serem usadas em uma rede, ou seja, cada rótulo representará uma diferente frequência. Como esse limite de frequências pode levar a uma solução não necessariamente conexa, pode-se conectar as componentes usando tecnologias "antigas". Pelo fato de a solução ser mínima em componentes, o custo e uso de conexões cabeadas são minimizados.

Aplicações como essa motivam o estudo do FGkR. Mesmo que restrições adicionais sejam impostas a esse tipo de aplicação, o problema em foco é básico para modelar cenários onde se deseja projetar alguma rede de comunicação/conexão.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo principal deste trabalho é estudar o problema FGkR, propor modelos de programação matemática e usar métodos exatos para resolver instâncias, com pelo menos 300 vértices, em tempo viável de computação.

Para atender esse objetivo tenta-se cumprir o seguinte:

- Revisar a bibliografia ligada ao tema, com ênfase em métodos exatos.
- Apresentar modelos matemáticos para o problema.
- Desenvolver métodos para solução exata desses modelos.
- Elaborar implementação paralela do método de *backtracking* existente na literatura.
- Comparar a melhor formulação com a literatura.

1.4 Estrutura do trabalho

No capitulo 2 são explorados fundamentos teóricos e metodológicos úteis para o desenvolvimento e entendimento do trabalho. São apresentadas propriedades básicas sobre componentes em grafos simples, o algoritmo de busca em profundidade, métodos exatos para resolução de modelos de programação inteira e uma revisão da literatura sobre grafos coloridos, com ênfase nos métodos necessários para este trabalho.

No Capítulo 3 são apresentadas caraterizações do problema em função de suas propriedades. É estudada a complexidade do problema, relacionando propriedades estruturais do grafo de entrada e o fator limitante k. Em particular, são identificados casos polinomiais.

No Capítulo 4 são propostos três modelos de programação inteira para o FGkR. São

exploradas modificações necessárias e melhorias para incrementar o desempenho do método de *Branch&Cut* usado para resolver os modelos. Ao final desse capítulo, são descritas as implementações usadas para resolver cada formulação bem como reportados alguns testes iniciais, melhor detalhados no apêndice A.

No Capítulo 5 são reportados os melhores ajustes feitos no *Branch&Cut* e no método de *backtracking*, baseados em testes com instâncias geradas de acordo com a literatura, porém em grafos com densidades maiores. Por fim, no Capítulo 6 são apresentadas as conclusões e discutidos alguns aspectos relevantes dos resultados obtidos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são discutidos aspectos pertinentes ao problema FGkR, dentre eles noções básicas da teoria de grafos como componentes e cortes, métodos para resolução de modelos de programação matemática e uma breve revisão da literatura.

2.1 Componentes e cortes de um grafo

Nesta seção são vistas algumas propriedades sobre número de componentes de um grafo, é apresentado um algoritmo de busca em profundidade para encontrar essa e outras informações de forma exata.

2.1.1 Propriedades Básicas

Relembre que uma componente de um GCA $G = (V,E,L,l)$ é um subgrafo conexo maximal (em vértices e arestas) $H = (V',E',l(E'),l)$. O número de vértices, arestas e componentes de G é representado por $n(G)$, $m(G)$ e $\mathscr{W}(G)$, respectivamente.

Veja que essa definição permite um particionamento dos vértices de G, ou seja, $V(G) = \bigcup_{j=1}^{\mathscr{W}(G)} W_j$ e $W_i \cap W_j = \emptyset$, para todo $1 \leq i < j \leq \mathscr{W}(G)$, onde W_j é o conjunto de vértices de G que estão na j-ésima componente.

Proposição 2.1.1. *Para todo GCA $G = (V,E,L,l)$ as seguintes desigualdades são válidas:*

1. $1 \leq \mathscr{W}(G) \leq n(G)$;
2. $\mathscr{W}(G) \leq \mathscr{W}(G[L']), \forall L' \subseteq L$.

Demonstração. A desigualdade 1 decorre do fato de que as componentes definem um particionamento de G. Como casos extremos, todos os vértices estão numa mesma partição ou cada vértice está numa partição diferente. Assim, $1 \leq \mathscr{W}(G) \leq n(G)$.

Para a desigualdade 2, lembre que, por definição, as arestas $E(L')$ de $G[L']$ estão contidas em G. Adicionando arestas de $E \backslash E(L')$ em $G[L']$, seu número de componentes só pode decrescer. Logo, $\mathscr{W}(G) \leq \mathscr{W}(G[L'])$. □

Como visto, ao adicionar uma aresta a um grafo pode-se manter o mesmo número de componentes ou gerar uma nova estrutura com exatamente uma componente a menos, caso a aresta acrescentada possua extremidades em diferentes componentes. Então, pode-se estabelecer as seguintes relações entre o número de vértices, arestas e componentes de um grafo.

Proposição 2.1.2. *Para todo grafo G,* $\mathscr{W}(G) \geq n(G) - m(G)$.

Demonstração. Um grafo G com n vértices e sem arestas possui n componentes (triviais). Ao adicionar m arestas iterativamente, cada uma pode conectar no máximo a um par de componentes disjuntas, reduzindo o número de componentes em no máximo 1 unidade. Logo, tem-se que $\mathscr{W}(G) \geq n(G) - m(G)$. □

Ao restringir a estrutura do grafo a uma floresta, o limitante da Proposição 2.1.2 é atingido.

Proposição 2.1.3. *Se F é uma floresta, então* $\mathscr{W}(F) = n(F) - m(F)$

Demonstração. Veja que, como uma floresta é um grafo com componentes minimalmente conexas, retirar uma aresta implica aumentar o número de componentes em exatamente uma unidade. Como o grafo resultante também é uma floresta, esse procedimento pode ser aplicado até termos $n(F)$ componentes, ou seja, até retirar todas as arestas do grafo. Então pode-se deduzir que o número de componentes da floresta original é $\mathscr{W}(F) = n(F) - m(F)$. □

O ato de retirar arestas de um grafo permite estudar propriedades interessantes em relação à sua conectividade. Quando uma aresta não pertence a um ciclo, sua remoção aumenta o número de componentes do grafo, e nesse caso é conhecida como aresta de corte ou ponte. Genericamente, um conjunto de arestas é dito desconectante quando sua exclusão do grafo aumenta o número de componentes. Os conjuntos desconectantes minimais podem ser obtidos de forma específica, como pode ser visto no algoritmo proposto em (STOER; WAGNER, 1997) para grafos não direcionados.

Definição 2.1.4. Sejam $G = (V, E)$ um grafo simples, S um subconjunto próprio não vazio de V e $\bar{S} = V - S$. O conjunto $\delta_G(S) = \{(i, j) \in E : i \in S, j \in \bar{S}\}$, também denotado $[S, \bar{S}]_G$, é dito um corte de arestas ou simplesmente corte.

Note que todo corte de arestas não vazio é um conjunto desconectante. Reciprocamente, todo corte desconectante minimal é um corte de arestas não vazio. Com essa definição, um grafo é conexo se, e somente se, todo corte de arestas for não vazio.

2.1.2 Algoritmo de Busca em Profundidade

O algoritmo de Busca em Profundidade (*Depth First Search* (DFS), em inglês) é um procedimento que visita os vértices de um grafo, segundo uma ordem específica, e recolhe

informações sobre sua estrutura. Intuitivamente, essa busca privilegia a visita de um vizinho do último vértice visitado, antes de retroceder para os outros vizinhos de um vértice visitado anteriormente.

Alguns exemplos de utilização desse procedimento podem ser vistos desde os anos 70 na área de inteligência artificial para busca em espaço de estados. Um dos seus principias usos consiste em encontrar as componentes de um grafo. Um pseudocódigo é exibido nos Algoritmos 1 e 2, baseado em (CORMEN *et al.*, 2009).

Algoritmo 1 Algoritmo DFS

```
1:  procedimento DFS(G)
2:      para u ∈ V(G) faça
3:          u.cor ← BRANCO
4:      fim para
5:      para u ∈ V(G) faça
6:          se u.cor == BRANCO então
7:              DFS_VISITA(G,u)
8:              u.π ← Nil
9:          fim se
10:     fim para
11: fim procedimento
```

Algoritmo 2 Algoritmo de visita DFS

```
1:  procedimento DFS_VISITA(G,u)
2:      u.cor ← CINZA
3:      para v ∈ 𝒩(u) faça
4:          se v.cor == BRANCO então
5:              v.π ← u
6:              DFS_VISITA(G,v)
7:          fim se
8:      fim para
9:      u.cor ← PRETO
10: fim procedimento
```

O algoritmo DFS (Algoritmo 1) inicializa, nas linhas 2-4, marcando todos os vértices de *G* como não explorados, ou seja, com o rótulo "BRANCO". Depois aplica o algoritmo de visita (DFS_VISITA) a partir dos vértices ainda não explorados, nas linhas 5-10. O algoritmo de visita (Algoritmo 2) muda o rótulo do vértice *u* corrente para "CINZA", na linha 2, e visita recursivamente todos os vértices com rótulo "BRANCO" na vizinhança $\mathcal{N}(u)$ de *u*, linhas 3-8. Quando toda vizinhança de *u* tiver sido explorada, esse vértice é marcado com o rótulo "PRETO",

linha 9.

O referencial $v.\pi$, determinado na linha 5, marca a partir de qual vértice v foi visitado, construindo-se assim uma árvore (de recursão) a partir de cada chamada ao algoritmo DFS_VISITA, na linha 7 do Algoritmo 1. Dessa forma, ao final da busca, é definida uma floresta geradora G_π, onde cada componente é definida por uma dessas árvores de recursão formada pelo Algoritmo 2.

Manter a informação em cada vértice v sobre quem é seu pai $v.\pi$ permite classificar as arestas do grafo em aresta de árvore ou aresta de retorno (CORMEN *et al.*, 2009). Dentro do laço que envolve as linhas 3-8 do Algoritmo 2, uma aresta (u,v) é classificada como de árvore quando v possui rótulo "BRANCO" e, portanto, faz parte da floresta G_π. Por outro lado, é classificada como de retorno quando v possui rótulo "CINZA", indicando que v é um "ancestral" de u em G_π. As arestas de retorno permitem identificar ciclos no grafo.

Veja que é possível contar o número de componentes de G trivialmente ao gerar G_π. O número de componentes é exatamente a quantidade de vezes que o teste da linha 6 do Algoritmo 1 é satisfeito.

A utilização de estruturas de dados auxiliares permite a fácil adaptação do algoritmo de visita para encontrar informações mais refinadas, por exemplo: tempo que o vértice foi visitado, uma estrutura de prioridade de visita para os vértices, ou até mesmo realizar operações com maior granularidade ao visitar um vértice. Mesmo assim, mantém-se a complexidade assintótica de tempo em $\Theta(|V|+|E|)$.

Quando é necessário adicionar repetidamente arestas ao grafo e verificar o número de componentes conexas, a estrutura de dados de conjuntos disjuntos (CORMEN *et al.*, 2009) pode ser indicada, por levar a uma melhor complexidade assintótica. A construção da estrutura é $O(|V|)$ e, para calcular o número de componentes de um grafo que é incrementado de E^* arestas, o algoritmo fica $O(|E^*|\alpha(|V|,|E^*|))$, onde $\alpha(|V|,|E^*|)$ é a função inversa de **Ackermann**, que na prática pode ser considerada constante (≤ 5).

2.2 *Branch&Bound*

Um Programa Linear Inteiro (PLI) é escrito da seguinte forma:

$$(PLI) \min \quad cx \tag{2.1a}$$

$$\text{s.a.} \quad Ax \leq b, \tag{2.1b}$$

$$x \geq 0, x \in \mathbb{Z}, \tag{2.1c}$$

onde $c = (c_1, c_2, c_3, .., c_n)$ é um vetor linha (dos custos), $A = [a_{ij}]$ é uma matriz $m \times n$ racional, b e x são vetores coluna, com $b^t = (b_1, b_2, .., b_m)$ e $x^t = (x_1, x_2, ..., x_n)$. O vetor x compreende as variáveis de decisão do problema. O conjunto $S = \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$ é dito conjunto de soluções viáveis do PLI, enquanto $f(x) = cx$ é sua função objetivo.

Quando as variáveis de decisão do problema são da forma $x \in \{0,1\}^n$, ou seja, binárias, chama-se esse problema de Programa Linear Binário (PLB). E caso somente um subconjunto das variáveis seja inteiras (e as demais reais), definindo o seguinte espaço de soluções viáveis $\{(x,y) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leq b\}$, o problema é dito Programa Linear Inteiro Misto (PLI), onde A,G e b são racionais. Quando se remove a restrição sobre o domínio das variáveis ser os inteiros, tem-se uma relaxação do PLI, o que define um Programa Linear (PL).

Sabe-se que um PL pode ser resolvido com algoritmos polinomiais (de pontos interiores). Todavia, embora tenha complexidade de pior caso exponencial, o algoritmo Simplex (DANTZIG, 1963) e (BAZARAA *et al.*, 2010), ainda é, possivelmente, o mais usado na prática. Apenas em exemplos raros (KLEE; MINTY, 1972), ele demanda um número exponencial de iterações.

Por outro lado, quando se introduz a necessidade de soluções com valores inteiros, métodos mais robustos são necessários, posto que os problemas do tipo PLI, PLI e PLB são NP-difíceis, em geral. Um desses métodos é o *Branch&Bound*, que consiste em relaxar as variáveis inteiras e utilizar um algoritmo de PL, como o Simplex, para encontrar uma solução. Se essa não for inteira, ele divide o problema misto em dois ou mais subproblemas com espaços de soluções viáveis complementares, de tal forma que, se uma solução ótima para o PLI existe, ela será a solução ótima de um dos subproblemas gerados. Essa mesma ideia pode ser aplicada aos subproblemas, gerando um processo recursivo, que pode ser representado por uma árvore (a árvore de *branch&bound*). Ao longo do método, podas são aplicadas para eliminar subproblemas cuja solução não seja, garantidamente, melhor que uma solução viável

já conhecida. O Algoritmo 3 apresenta um pseudocódigo do método, aplicado a um PLI (de minimização):

Algoritmo 3 Método B&B

1: **procedimento** B&B(P_0)
2: Criar nó N_0 associado a P_0
3: $\mathscr{L} = \{N_0\}$
4: $\bar{z} \leftarrow +\infty$, $(x^*, y^*) \leftarrow \emptyset$
5: **se** $\mathscr{L} == \emptyset$ **então**
6: retornar (x^*, y^*) como ótimo.
7: **fim se**
8: Selecionar um nó N_i de $\mathscr{L}$
9: remover N_i de $\mathscr{L}$
10: resolver relaxação PL_i de P_i
11: **se** PL_i for inviável **então**
12: retornar para linha 5
13: **fim se**
14: (x^i, y^i) é a solução de LP_i e z_i seu valor objetivo
15: **se** $z_i \geq \bar{z}$ **então**
16: retornar para linha 5
17: **fim se**
18: **se** (x^i, y^i) for viável para o PLIM **então**
19: $\bar{z} \leftarrow z_i$
20: $(x^*, y^*) \leftarrow (x^i, y^i)$
21: retornar para linha 5
22: **fim se**
23: A partir de P_i criar $k \geq 2$ subproblemas $P_{i_1}, ..., P_{i_k}$
24: Adicionar a $\mathscr{L}$ os novos nós $N_{i_1}, ..., N_{i_k}$ correspondentes
25: retornar linha 5
26: **fim procedimento**

No Algoritmo 3, começando de um nó N_0, que representa o PLI P_0 a ser resolvido, tem-se como objetivo gerar uma solução (x^*, y^*) ótima com custo $\bar{z}$, cujos valores são inicializados com $\emptyset$ e $+\infty$ na linha 4. Os nós que serão explorados serão mantidos em uma lista $\mathscr{L}$ e, ao esvaziar essa lista, a melhor solução encontrada é retornada (linhas 5-7). Iterativamente, escolhe-se e remove-se dessa lista um nó a ser processado, linhas 8-9. Vale ressalvar que existem várias formas de escolher um nó e que esse processo causa impacto no desempenho do algoritmo.

O processamento de um nó envolve algumas etapas. Primeiro, resolve-se sua relaxação linear (linha 10), cujo valor fornece um limite inferior (*bound*) para o subproblema corrente. Depois que o problema relaxado é resolvido, testes de poda são executados (linhas 11-22) para evitar ter que resolver desnecessariamente os possíveis descendentes da subárvore

de busca enraizada no nó atual. Três casos de poda podem ser encontrados:

- Poda por inviabilidade, que ocorre quando PL_i é inviável. Logo o subproblema P_i não pode conter uma solução ótima de P_0.
- Poda por limite, que ocorre quando o valor ótimo z_i da relaxação do subproblema corrente é maior que ou igual ao valor $\bar{z}$ da melhor solução encontrada até o momento. Logo, o valor de uma solução ótima de P_i, que é pelo menos z_i, não pode ser melhor que $\bar{z}$, pois o PLI é um problema de minimização. Em outras palavras, não existem candidatos à solução de P_0 com valor melhor que $\bar{z}$ entre as soluções viáveis de P_i.
- Poda por otimalidade, que ocorre quando a solução (x^i, y^i) de LP_i for uma solução viável para o PLI e, portanto, ótima para P_i. Tendo passado pelo teste de poda por limite sem sucesso, isto significa também que (x^i, y^i) for uma solução viável de P_0 melhor que a atualmente conhecida.

Vale mencionar ainda que na resolução de LP_0 pode acontecer de a relaxação ser um problema ilimitado. Nesse caso, pode-se mostrar que P_0 também é ilimitado e o algoritmo já pode parar na primeira iteração.

O processo de ramificação de N_i nas linhas 23-24 consiste em criar subproblemas de tal forma que o conjunto de soluções viáveis do PLI contidas em P_i também está contido na união dos espaços de soluções viáveis desses subproblemas. Existem diferentes formas de fazer esse processo de ramificação (*branching*). Dentre as mais conhecidas, encontra-se o procedimento de ramificação em variáveis.

Por exemplo, sejam S o espaço de soluções viáveis de um PLI P e (x^*, y^*) a solução ótima de sua relaxação linear, onde uma componente x_i^* possui um valor fracionário f (caso contrário teria havido uma poda). Veja que, para $S_1 = \{(x,y) \in S : x_i \leq \lfloor f \rfloor\}$ e $S_2 = \{(x,y) \in S : x_i \geq \lceil f \rceil\}$, $S = S_1 \cup S_2$ é válida. Dessa maneira, podemos definir subproblemas P_1 e P_2 a partir de S_1 e S_2, respectivamente. Note que (x^*, y^*) não é mais viável para qualquer dos subproblemas.

2.3 Planos de Corte

O algoritmo de planos de corte é outro método exato usado na resolução de um PLI. Resolvida a relaxação linear do PLI, caso ela não seja inteira, a ideia é introduzir desigualdades para eliminar essa solução, sem descartar qualquer solução viável do PLI, fortalecendo assim a relaxação do problema. Isso feito, repete-se o processo a partir na nova relaxação.

Para formalizar essa ideia, considere $S = \{(x,y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^p : Ax + Gy \leq b\}$, com

A,G e b racionais, o conjunto de soluções viáveis de um PLI P. Seja P' a relaxação linear de P, cujo conjunto viável é $S' = \{(x,y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leq b\}$.

Uma desigualdade $\alpha x + \gamma y \leq \beta$ é uma desigualdade válida para $S \subseteq \mathbb{R}^n_+ \times \mathbb{R}^p_+$, se $\alpha x + \gamma y \leq \beta$ para todo $(x,y) \in S$. Se (x',y') for uma solução ótima de P' tal que $(x',y') \notin S$, então uma desigualdade válida $\alpha x + \gamma y \leq \beta$ para S tal que $\alpha x^i + \gamma y^i > \beta$ define um plano de corte que separa (x',y') de S.

Dado um plano de corte $\alpha x + \gamma y \leq \beta$, define-se uma nova relaxação P'' a partir do conjunto

$$S'' = S' \cap \{(x,y) : \alpha x + \gamma y \leq \beta\}.$$

Veja que P'' é mais forte que P' pois $S \subseteq S'' \subset S' \setminus \{(x',y')\}$. O procedimento de planos de corte consiste em repetir esse processo recursivamente, introduzindo na relaxação corrente uma ou mais desigualdades válidas, até que sua solução ótima seja viável para P.

Encontrar desigualdades válidas para S que o separem de um ponto (x',y') define um problema de separação. A qualidade e quantidade de cortes encontrados na separação e a estratégia de inclusão dos mesmos são o que definem o custo operacional e a eficácia do procedimento de planos de corte. Esses aspectos são ainda mais relevantes quando combina-se os métodos de *Branch&Bound* e planos de corte, definindo o que se chama *Branch&Cut*. O método *Branch&Cut* consiste em acoplar um ou mais planos de corte para fortalecer a relaxação, antes de fazer uma ramificação.

2.4 Decomposição de Benders

A decomposição de Benders no contexto de um PLI consiste na separação da formulação em um problema mestre e um ou mais escravos. O problema mestre pode ser visto como uma reformulação do problema original, enquanto os subproblemas escravos são responsáveis por gerar as restrições do mestre. Esse procedimento é interessante quando se deseja resolver PLI's com número grande de variáveis contínuas.

Seja um PLI da forma (2.2a-e), onde x e y são as variáveis de decisão (inteiras e contínuas, respectivamente), f e c são vetores linha racionais que definem a função objetivo, e A,B,D, b e d são matrizes e vetores coluna racionais que expressam as restrições.

$$(PLIM1)\ \min_{x,y} \quad fy + cx \tag{2.2a}$$

$$\text{s.a.} \quad Ay \leq b, \tag{2.2b}$$

$$By + Dx = d, \tag{2.2c}$$

$$x \geq 0, \tag{2.2d}$$

$$y \geq 0,\ y \in \mathbb{Z}^m. \tag{2.2e}$$

Por simplicidade, admita que a relaxação linear de PLIM1 não seja ilimitada, caso contrário esse problema é ilimitado. Sendo $Y = \{y \in \mathbb{Z}^m_+ : Ay \leq b\}$, considere a seguinte transformação de PLIM1:

$$(PLIM1.1)\ \min_{y,z} \quad fy + z \tag{2.3a}$$

$$\text{s.a.} \quad z = \min_x \{cx : Dx = d - By, x \geq 0\}, \tag{2.3b}$$

$$y \in Y. \tag{2.3c}$$

onde $z = z(y)$ é o valor ótimo do problema (parametrizado em y) expresso em (2.3b). Em particular, $z = +\infty$ ou $z = -\infty$, caso esse problema seja inviável ou ilimitado, respectivamente.

Como justificado a seguir, o subproblema (2.3b) pode ser substituído pelo seu dual:

$$(D_y)\ \max_{\mu} \quad \mu(d - By) \tag{2.4a}$$

$$\text{s.a.} \quad \mu D \leq c^T \tag{2.4b}$$

O espaço de soluções viáveis de (D_y), $F = \{\mu : \mu D \leq c^T\}$, é independente de y. Caso seja vazio, temos que, para todo y, (2.3b) define um problema inviável (não pode ser ilimitado pois a relaxação de PLIM1 não é ilimitada). Nesse caso, PLIM1 também é inviável. Portanto, pode-se admitir que F é não vazio. Sejam $\{r^j : j \in J\}$ o conjunto de direções extremas de F e $\{\mu^k : k \in K\} \neq \emptyset$ o conjunto de pontos extremos (vértices) de F.

Sendo $F \neq \emptyset$, o problema (D_y) é ilimitado ou possui uma solução ótima básica $\bar{\mu}$. Suponha que (D_y) seja ilimitado e, portanto, y não define solução viável para PLIM1 ($z(y) = +\infty$). Logo, existe uma direção extrema r^j, $j \in J$, tal que $r^j(d - By) > 0$. Assim, para eliminar todo

$y \in Y$ tal que $z(y) = +\infty$, os seguintes cortes são utilizados:

$$r^j(d - By) \leq 0, \quad \forall j \in J \tag{2.5}$$

Por outro lado, se (D_y) possui solução ótima, essa é do tipo μ^k, $k \in K$. Em outras palavras, se $z(y)$ é finito, então $z(y) = \max\{\mu^k(d - By) : k \in K\}$.

Usando os argumentos apresentados acima, pode-se criar o seguinte problema mestre, equivalente a PLIM1:

$$(PLIM2 - MP) \min_{y,z} \quad fy + z \tag{2.6a}$$

$$\text{s.a.} \qquad z \geq \mu^k(d - By) \ \forall k \in K, \tag{2.6b}$$

$$r^j(d - By) \leq 0 \qquad \forall j \in J, \tag{2.6c}$$

$$y \in Y. \tag{2.6d}$$

No modelo acima, todas as variáveis, à exceção de z, são inteiras. Entretanto, essa reformulação, proposta por (BENDERS, 1962), leva a um número potencialmente exponencial de restrições do tipo (2.6a) e (2.6b). O método de planos de corte pode então ser aplicado. Começando com um subconjunto das desigualdades, obtém-se uma solução ótima $(\bar{y}, \bar{z})$ do problema mestre corrente. Então resolve-se $(D_{\bar{y}})$. Se for ilimitado, é encontrada uma direção extrema $\bar{r}$ satisfazendo $\bar{r}(d - B\bar{y}) > 0$, e uma desigualdade do tipo (2.6c) é acrescentada, cortando $(\bar{y}, \bar{z})$. Caso contrário, $(D_{\bar{y}})$ tem solução ótima $\bar{\mu}$. Se $\bar{z} < \bar{\mu}(d - B\bar{y})$, então uma desigualdade do tipo (2.6c) é adicionada, cortando $(\bar{y}, \bar{z})$. Em último caso, tem-se que $\bar{z} \geq \bar{\mu}(d - B\bar{y}) = \max\{\mu^k(d - B\bar{y}) : k \in K\}$ e $r^j(d - B\bar{y}) \leq 0$ para todo $j \in J$, mostrando que $(\bar{y}, \bar{z})$ é solução ótima.

2.5 Revisão Bibliográfica sobre o FGkR

O problema da Floresta Geradora k-Rotulada foi estabelecido recentemente em (CERULLI *et al.*, 2014). A literatura associado ao problema ainda é bastante restrita. Nesta seção, são explorados métodos da literatura que são aplicáveis ao problema e estão relacionados ao escopo deste trabalho.

*2.5.1 Heurística k**MVCA***

O kMVCA consiste de uma adaptação simples da heurística gulosa *Maximum Vertex Covering Algorithm* (MVCA), proposta para o MLSTP. A MVCA, inicialmente sugerida por

(CHANG; SHING-JIUAN, 1997) e depois revisada em (KRUMKE; WIRTH, 1998), constrói um subgrafo gerador conexo, acrescentando arestas iterativamente e procurando minimizar o número de rótulos usados. A adaptação para o FGkR, sugerida por (CERULLI *et al.*, 2014), essencialmente abrevia o término da heurística, que passa a ocorrer quando um número k de rótulos é atingido. Para melhor entendimento, considere que um vértice está coberto quando for extremidade de uma das arestas acrescentadas no processo iterativo.

Mais precisamente, na versão original do MVCA proposta por (CHANG; SHING-JIUAN, 1997), aplicada a um GCA $G = (V, E, L, l)$, cria-se inicialmente um grafo $G^* = (V, E^* = \emptyset, L^* = \emptyset, l)$. A partir desse grafo, escolhe-se iterativamente um rótulo $l^* \in L - L^*$ cujas arestas com ele rotuladas cubram a maior quantidade de vértices ainda não cobertos. Inclui-se l^* em L^* e as arestas correspondentes em E^*, e o processo é repetido até G^* ser conexo. Quando dois rótulos cobrem a mesma quantidade de vértices, escolhe-se um deles de forma arbitrária. Em (KRUMKE; WIRTH, 1998) é mostrada uma versão revisada, que é usada neste trabalho.

Na versão revisada por (KRUMKE; WIRTH, 1998), a escolha do rótulo $l^* \in L - L^*$ a ser acrescido a L^* é tal que minimiza o número de componentes de G^* com a inclusão das arestas de rótulo l^*. A complexidade de tempo desse algoritmo é $O(|E| + |L| \cdot \alpha(|E|, |V|) \cdot \min\{|L| \cdot |V|, |E|\})$, onde α é a função inversa de **Ackermann**, que para fins práticos é no máximo 5. (KRUMKE; WIRTH, 1998) mostram um fator de aproximação para o MVCA de $1 + 2ln|V|$ vezes o ótimo, que é melhorado em (WAN *et al.*, 2002) para $1 + ln(|V| - 1)$. Em (XIONG *et al.*, 2005) um limitante melhor é determinado para o fator de aproximação. Sendo $b \in \mathbb{Z}^+$ um limite superior (justo) para o número de arestas por cor, esse fator de aproximação é dado pela média harmônica $H_b = \sum_{i=1}^{b} i^{-1}$.

O que (CERULLI *et al.*, 2014) propõem é simplesmente aplicar o procedimento de (KRUMKE; WIRTH, 1998), limitando o número de rótulos usados a k, como pode ser visto no Algoritmo 4.

Algoritmo 4 Algoritmo kMVCA

procedimento kMVCA(G,k)
  $E^*, L^* \leftarrow \emptyset$
  $G^* \leftarrow (V, E^*, L^*, l)$
  enquanto $(\mathscr{W}(G^*) > 1)$ & $(|L^*| < k)$ **faça**
    $c^* \leftarrow \arg\min\{\mathscr{W}(G[L^* \cup c]) : c \in L - L^*\}$
    $L^* \leftarrow L^* \cup \{c^*\}$
    $G^* \leftarrow G[L^*]$
  fim enquanto
  gerar floresta geradora de G^*
fim procedimento

Vale ressaltar que (SILVA, 2018) propõe uma versão revisada rMVCA do MVCA, com complexidade $O(\alpha(|V|,|V|) \cdot |L| \cdot |V|)$, para o MLSTP. Ele consegue esse resultado através de uma nova implementação do MVCA com a utilização de múltiplas estruturas de conjuntos disjuntos, uma para cada cor ainda não escolhida, e de operações de contração sobre o grafo. A estrutura referente a uma cor representa as componentes conexas do subgrafo induzido por essa cor e as cores já escolhidas. Cada componente é contraída em um único vértice, de modo a reduzir o número de operações sobre os conjuntos até encontrar um solução conexa (um vértice depois da contração). A implementação usada em rMVCA também se aplica a kMVCA.

2.5.2 Algoritmo Enumerativo

O algoritmo exato proposto por (CERULLI *et al.*, 2014) aplica um procedimento de *backtracking*. Ele se baseia no fato de que o problema FGkR, conforme Proposição 1.2.2, consiste em determinar

$$\min\{\mathscr{W}(G[L']) : L' \subseteq L, |L'| \leq k\}.$$

Com essa perspectiva, o procedimento enumera implicitamente todos os subconjuntos de rótulos $L' \subseteq L$, com $|L'| \leq k$, e guarda o que fornece o menor valor de $\mathscr{W}(G[L'])$. Esse processo é organizado em uma árvore de enumeração, como no método *Branch&Bound*, onde a ramificação é feita escolhendo-se um rótulo para ser usado ou proibido na solução, e um limitante é obtido com o uso da seguinte proposição, como é explicado a seguir.

Proposição 2.5.1. *(CERULLI et al., 2014) Seja um GCA $G = (V, E, L, l)$ e dois subconjuntos não vazios $M, M' \subseteq L$ tais que $M \cap M' = \emptyset$ e $M' \geq 1$. A seguinte desigualdade é válida:*

$$\mathscr{W}(G[M]) - \mathscr{W}(G[M \cup M']) \leq \sum_{c \in M'} (\mathscr{W}(G[M]) - \mathscr{W}(G[M \cup \{c\}])). \tag{2.7}$$

Cada nó da árvore de enumeração é representado por uma solução parcial, definida por um conjunto de rótulos já escolhidos e outros ainda candidatos. Na verdade, é estabelecida uma ordem inicial dos rótulos $L = \{c_1, c_2, \ldots, c_{|L|}\}$ e, se d é a profundidade do nó, então os rótulos escolhidos em sua solução parcial formam um subconjunto de $\{c_1, \ldots, c_d\}$, enquanto os candidatos são os rótulos $c_{d+1}, \ldots, c_{|L|}$. Os rótulos são ordenados em ordem decrescente da quantidade de arestas de cada um, ou seja, $|E(c_i)| \geq |E(c_{i+1})|$ para todo $i \in \{1, \ldots, |L| - 1\}$.

O Algoritmo 5 apresenta um pseudocódigo desse procedimento, onde L' representa a solução parcial (rótulos selecionados) associada ao nó corrente, L^* é a melhor solução conhecida e d a profundidade do nó na árvore. Inicialmente, $L' = \emptyset$, $L^* = \emptyset$ e $d = 0$.

Algoritmo 5 Algoritmo Exato FGkR

```
1:  procedimento Branch&Bound(L',d,L*)
2:      se |L'| == k então
3:          se W(G[L']) < W(G[L*]) então
4:              L* ← L'
5:              se W(G[L*]) == 1 então
6:                  retornar L* e parar algoritmo
7:              fim se
8:              retornar L*
9:          fim se
10:     fim se
11:     se |L| − d < k − |L'| então
12:         retornar L*
13:     fim se
14:     atribuir a maxReduction o valor da maior redução possível para W(G[L'])
15:     se W(G[L']) − maxReduction ≥ W(G[L*]) então
16:         retornar L*
17:     fim se
18:     L' ← L' ∪ {c_{d+1}}
19:     L* ← Branch&Bound(L', d + 1, L*)
20:     L' ← L' \ {c_{d+1}}
21:     L* ← Branch&Bound(L', d + 1, L*)
22:     retornar L*
23: fim procedimento
```

Começando da solução inicial, cada nó da árvore de busca é ramificado (*branching*), com base em sua profundidade d, em dois casos, gerando dois nós filhos (linhas 19 e 20):

- Em um caso, a solução parcial L' será atualizada com proibição do rótulo c_{d+1} (linha 20).
- No outro caso, a solução parcial L' será atualizada com a escolha do rótulo c_{d+1} (linha 18), obrigando-o a fazer parte das soluções enraizadas nesse nó.

Durante o processo de chamadas recursivas, ao encontrar uma solução parcial com k cores (linha 2), essa será comparada (linha 3) com a melhor solução até o momento, armazenada em L^*, e, caso melhor, atualiza-se a melhor solução encontrada (linha 4). No caso de esta ser a solução trivial ($\mathscr{W}(G[L'])$ igual a 1), a solução ótima foi encontrada, e o processo termina (linha 6). Caso contrário, simplesmente retorna-se o novo L^* (linha 8).

Quando a solução parcial associada ao nó corrente (de profundidade d) possuir menos que k cores já escolhidas, ou seja, $|L'| < k$, é verificado se ainda é possível criar uma solução com exatamente k cores (linha 11). Caso contrário, uma poda por inviabilidade é aplicada.

Além disso, (CERULLI *et al.*, 2014) propõem o uso de limite inferior (*lower bound*), derivado da Proposição 2.5.1, para podar soluções parciais que não possam gerar soluções com valor melhor que o limite superior ($\mathscr{W}(G[L^*])$) do nó atual (linha 14).

A Proposição 2.5.1 é empregada no Algoritmo 5 de forma que $M = L'$, $M' \subseteq \{c_{d+1}, c_{d+2}, ..., c_{|L|}\}$, $|M' \cup M| = k$ e M' geraria a maior redução possível para $\mathscr{W}(G[M])$ (*maxReduction*), ou seja, o maior valor possível para o lado direito da desigualdade na Proposição 2.5.1. Precisamente,

$$maxReduction = \max\left\{ \sum_{c \in M'} (\mathscr{W}(G[L']) - \mathscr{W}(G[L' \cup \{c\}]) : M' \subseteq \{c_{d+1}, ..., c_{|L|}\}, |M'| = k - |L'|) \right\} \tag{2.8}$$

O algoritmo que (CERULLI *et al.*, 2014) propõem para calcular *maxReduction* consiste em acumular a diferença $\mathscr{W}(G[L']) - \mathscr{W}(G[L' \cup \{c\}]$ para cada cor c que ainda não foi escolhida, usando estrutura de dados para determinar o quanto o conjunto de arestas da cor c contribui para reduzir o número de componentes de $\mathscr{W}(G[L'])$. Esse procedimento é feito de forma otimizada, analisando as arestas que estão entre componentes de $G[L^{'}]$ e mantendo nas estruturas como essas arestas conectam as componentes.

Vale ressalvar que o algoritmo apresenta resultados exatos em tempo viável de até 3 horas para $200 \sim 300$ vértices (CONSOLI *et al.*, 2017).

3 CARACTERIZAÇÕES DO PROBLEMA

Neste capítulo são estudadas características de um GCA que possuem relação com o problema. Em particular, são apresentados diversos conceitos que fundamentam estratégias de solução, tais como componentes monocromáticas, cortes coloridos, cintura colorida, dominância de cores e contração de cores. Adicionalmente, são estabelecidos problemas equivalentes, que são essenciais para o estudo do FGkR e criação de formulações matemáticas.

3.1 Propriedades sobre um GCA

A Proposição 1.2.2 estabelece que resolver o FGkR consiste em encontrar um subconjunto com k rótulos que induz um subgrafo gerador com número mínimo de componentes. Para tanto vale estudar algumas características de um GCA com relação ao problema.

3.1.1 Componentes Monocromáticas

Um GCA $G = (V, E, L, l)$ é dito monocromático se todas as suas arestas possuem a mesma cor, ou seja, $|l(E)| = 1$. Para cada $l \in L$, $G[\{l\}]$ é, portanto, um subgrafo monocromático (induzido por l).

Definição 3.1.1. Seja $G = (V, E, L, l)$ um GCA. O conjunto de vértices de cada componente conexa de um subgrafo monocromático $G[\{l\}]$, para qualquer $l \in L$, é dito uma componente monocromática de G.

Note que componentes monocromáticas relativas a uma mesma cor são disjuntas, porém podem se interceptar quando geradas por cores distintas. O número de componentes monocromáticas é $\sum_{l \in L} \mathscr{W}(G[\{l\}])$.

À luz da Proposição 1.2.2, uma solução para o FGkR pode ser vista como a união de componentes monocromáticas de G. Por outro lado, se arestas fossem removidas dessas componentes de modo a torná-las acíclicas, a floresta obtida também representaria a mesma solução ótima. Nesse sentido, percebe-se que modificações podem ser feitas no grafo original sem alterar a solução ótima do problema, como se detalha a seguir.

Definição 3.1.2. Um ciclo monocromático C em um GCA $G = (V, E, L, l)$ é um ciclo tal que $|l(E(C))| = 1$.

A seguinte propriedade é uma adaptação daquela usada em (SILVA, 2018) no contexto do MLSTP.

Proposição 3.1.3. *Se $G' \subseteq G$ é um subgrafo gerador livre de ciclo monocromático, maximal em arestas, então toda solução ótima para o FGkR em G' é ótima para o FGkR em G.*

Demonstração. Observe que G e G' possuem as mesmas componentes monocromáticas, pois as arestas em $E(G) \setminus E(G')$ apenas quebram os ciclos monocromáticos de G. Consequentemente, $\mathscr{W}(G[L']) = \mathscr{W}(G'[L'])$, para todo $L' \subseteq L$. O resultado então segue pela Proposição 1.2.2. □

Na verdade, o resultado acima pode ser estendido, como segue.

Proposição 3.1.4. *Se G e G' são dois GCAs com as mesmas componentes monocromáticas, então o problema FGkR em G e G' são equivalentes.*

Demonstração. Como G e G' possuem as mesmas componentes monocromáticas, $\mathscr{W}(G[L']) = \mathscr{W}(G'[L'])$, para todo $L' \subseteq L$. Então a equivalência decorre da Proposição 1.2.2. □

Pela proposição acima, contanto que as componentes monocromáticas sejam preservadas, é possível criar e remover arestas em G, restritas a uma mesma componente monocromáticas, sem mudar o problema. A Figura 5 ilustra algumas dessas possíveis transformações. Considere que uma componente monocromática (vermelha) de G seja um ciclo com 5 vértices. Uma mudança possível em G seria remover uma das arestas do ciclo. Um segunda alternativa seria adicionar arestas (vermelhas) entre todos os vértices da componente, gerando um subgrafo completo monocromático. Uma terceira seria definir uma estrela, centrada em um dos vértices do ciclo (nesse caso, tanto se remove quanto se adiciona arestas vermelhas). Em geral, a componente poderia ser substituída por qualquer subgrafo conexo envolvendo esses cinco vértices.

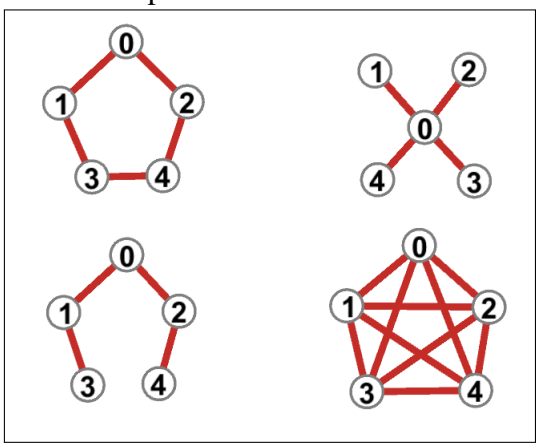
Quando tais modificações são feitas, veja que podem aparecer arestas paralelas, de cores diferentes, entre um mesmo par de vértices, transformando assim o grafo num multigrafo.

3.1.2 Cortes Coloridos

Com respeito a cortes em grafos coloridos, (SILVA *et al.*, 2019) definem os cortes coloridos. Eles fornecem propriedades inerentes aos modelos que serão definidos.

Definição 3.1.5. Sejam $G = (V, E, L, l)$ um GCA e S um subconjunto próprio não vazio de V. Denomina-se $K_G(S) = l(\delta_G(S))$ um corte colorido.

Figura 5 – Transformações em Componentes Monocromáticas.



Fonte: elaborado pelo autor (2020).

Quando o grafo G está claro pelo contexto, é possível remover o sobrescrito $_G$ das notações $\delta_G(S)$ e $K_G(S)$.

A Definição 3.1.5 conecta-se a uma propriedade muito importante neste trabalho.

Proposição 3.1.6. *(SILVA, 2018) Sejam $G = (V,E,L,l)$ um GCA e $H = (V,E',l(E'),l)$ um subgrafo gerador de G. Então H é conexo se, e somente se, $l(E')$ contém pelo menos uma cor de todo corte colorido não-vazio de G.*

Demonstração. Seja $H = (V,E',l(E'),l)$ um subgrafo gerador de G. Suponha que H é conexo e seja $K_G(S) = l(\delta_G(S))$ um corte colorido não vazio de G. Então, $\delta_G(S) \neq \emptyset$. Como H é gerador e conexo, $\delta_H(S) = \delta_G(S) \cap E'$ é um corte não vazio de H. Como $l(E')$ são os rótulos das arestas em E', então $\emptyset \neq l(\delta_G(S) \cap E') \subseteq l(E')$. Portanto, $l(\delta_G(S)) \cap l(E') \neq \emptyset$.

Suponha agora que H seja desconexo. Então existe corte vazio $\delta_H(S)$. Logo, toda possível aresta em $\delta_G(S)$ tem cor em $L \setminus l(E')$. Em outros termos, $l(E') \cap K_G(S) = \emptyset$. □

3.1.3 Dominância e Contração

Nesta subseção são apresentados os conceitos de dominância e contração de cor em um GCA, introduzidos em (SILVA, 2018).

Definição 3.1.7. Dados um subconjunto de rótulos não vazio $L' \subset L$ e um rótulo $c \in L \setminus L'$, c é dito dominado por L', se $\mathscr{W}(G[L']) = \mathscr{W}(G[L' \cup \{c\}])$.

Definição 3.1.8. A contração da aresta $e = (x,y) \in E$ em um GCA $G = (V,E,L,l)$ consiste na remoção desta aresta e na junção de x e y em um novo vértice v, que passa então a ser extremidade das arestas que incidiam em x ou y, obtendo assim um multigrafo colorido em arestas, a ser

denotado $G \mid e$. A contração de um subconjunto de arestas $E' \subset E$ é a aplicação recursiva da contração de cada aresta em E', ou seja, $G \mid E' = (G \mid E'') \mid e$, onde $E'' = E' \setminus \{e\}$ para algum $e \in E'$.

Definição 3.1.9. A operação de contração de um subconjunto de rótulos $L' \subset L$ em um GCA $G = (V, E, L, l)$ é dada pela contração do conjunto de arestas $E(L') = \{e \in E : l(e) \in L'\}$ em G, levando a um multigrafo, denotado por $G \parallel L'$.

Uma estratégia para resolver o FGkR consiste em construir iterativamente o conjunto de rótulos a serem escolhidos. Dessa forma, uma solução parcial para o problema é qualquer subconjunto com até k rótulos. Observe que, se L' é uma solução parcial e $c \in L \setminus L'$ é dominada por L', então c pode ser descartada na formação da solução, pois não seria útil para reduzir o número de componentes (só seria talvez usada para completar exatamente k cores, caso fosse considerada essa definição do problema).

Segundo (SILVA, 2018), a contração de rótulos que fazem parte de uma solução parcial é útil em dois sentidos. Primeiro, permite a verificação de quais rótulos ainda candidatos a entrarem na solução são dominados e que, portanto, podem ser descartados. Os rótulos dominados são aqueles que colorem apenas laços (arestas com extremidades iguais) do multigrafo contraído. Segundo, essa operação também permite trabalhar com um multigrafo que possui menos vértices e arestas que o grafo original, de forma equivalente. O número de vértices é naturalmente reduzido pela junção das extremidades das arestas contraídas. E além dessas arestas serem removidas, todo laço formado (e não apenas aqueles relativos a rótulos dominados) também pode ser eliminado. De fato, um laço representa uma aresta do grafo original que liga dois vértices de uma componente já formada pelos rótulos da solução parcial. Observe que a eliminação dos laços naturalmente exclui os rótulos dominados, que deixam de aparecer no multigrafo contraído. Vale salientar a adequação das estruturas de conjuntos disjuntos para implementar processo de contração.

Destaca-se a seguir a forte relação entre o número de vértices do multigrafo contraído e o cálculo do limitante inferior usado no Algoritmo 5, linhas 14-15. De fato, note que, dado um subconjunto $L' \subseteq L$, cada componente conexa de $G[L']$ se transforma em um vértice de $G \parallel L'$, de modo que $\mathscr{W}(G[L']) = n(G \parallel L')$. Então, os coeficientes da expressão (2.8) podem ser dados por $\mathscr{W}(G[L']) - \mathscr{W}(G[L' \cup \{c\}]) = n(G \parallel L') - n((G \parallel L') \parallel c)$.

3.2 Problemas equivalentes

Para construir modelos matemáticos para o problema da FGkR, apresentamos primeiro diferentes caracterizações de soluções ótimas. Dado que $\mathscr{F}_G(k)$ é o conjunto de todas as florestas geradoras de G com no máximo k rótulos distintos, ou seja, $\mathscr{F}_G(k)$ é o conjunto de todas as soluções válidas para o problema.

Proposição 3.2.1. *Sejam $G = (V,E,L,l)$ um GCA e $k \in \mathbb{Z}^+$. Então $F^* \in \mathscr{F}_G(k)$ é uma solução ótima para FGkR em G se, e somente se, $|E(F^*)| \geq |E(F)|, \forall F \in \mathscr{F}_G(k)$.*

Demonstração. A floresta F^* é uma solução ótima se, e somente se, $\mathscr{W}(F^*) = \min\{\mathscr{W}(F) \mid F \in \mathscr{F}_G(k)\}$. Veja que, pela Proposição 2.1.3, $\mathscr{W}(F^*) = n(G) - |E(F^*)|$. Portanto, minimizar o número de componentes é equivalente a maximizar o número de arestas, dado que $n(G)$ é fixo. Logo, F^* é máxima em arestas em relação às florestas em $\mathscr{F}_G(k)$. □

Essa proposição é interessante, no sentido de que verificar o número de componentes de elementos de $\mathscr{F}_G(k)$ é mais custoso do que simplesmente analisar o número de arestas. Porém, veja que a Proposição 3.2.1 só vale quando restringimos as soluções viáveis de FGkR a florestas (em princípio, pela Proposição 1.2.2, podem-se usar soluções relaxadas equivalentes para compreenderem todos os subgrafos geradores com até k componentes). Um modelo baseado nessa proposição deve impor então restrições de quebra de ciclos.

As cores usadas em uma solução são o que define as arestas que podem ser utilizadas na mesma. O problema da FGkR pode então ser expresso como aquele de encontrar um subconjunto de rótulos $L' \subseteq L$, $|L'| \leq k$, cujo subgrafo induzido $G[L']$ possui o menor número de componentes. Tal subgrafo gerador não necessariamente é uma floresta, mas existe algoritmo com complexidade de tempo polinomial que permite converter essa solução para uma floresta geradora com a mesma quantidade de componentes.

Nesse caso, porém, o valor objetivo não pode ser calculado a partir do número de arestas do subgrafo gerador obtido. Para determinar o número de componentes através de uma função linear simples, converte-se o FGkR, definido sobre um GCA G, em um MLSTP equivalente definido sobre um grafo estendido G', como segue.

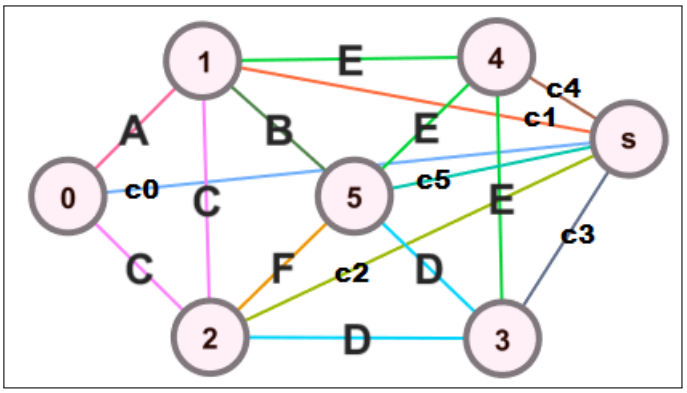
Definição 3.2.2. Dado um GCA $G = (V,E,L,l)$, chamamos de GCA estendido o grafo $G' = (V',E',L',l')$ conexo, onde $V' = V \cup \{s\}$ tal que s é um novo vértice, $E' = E \cup \{(s,v) : v \in V\}$, $L' = L \cup C$, sendo $C = \{c_v : v \in V\}$ um conjunto de novos rótulos distintos ($C \cap L = \emptyset$), e l' é tal

que:

$$l'(e) = \begin{cases} l(e), & e \in E, \\ c_v, & e = (s,v), v \in V. \end{cases} \tag{3.1}$$

A Figura 6 apresenta o grafo estendido associado ao GCA da Figura 2.

Figura 6 – Grafo G'



Fonte: elaborado pelo autor (2020).

Proposição 3.2.3. *O problema FGkR em G é equivalente a encontrar um subgrafo gerador conexo $G'[L^* \cup C^*]$, com $L^* \subseteq L$ e $C^* \subseteq C$, tal que $|L^*| \leq k$ e $|C^*|$ seja mínimo.*

Demonstração. Seja $L^* \subseteq L$ um subconjunto de rótulos. Sejam $U \subseteq V$ um conjunto com exatamente um vértice de cada componente conexa de $G[L^*]$ e $C^* = \{c_v : v \in U\}$. Temos que $G'[L^* \cup C^*]$ é um subgrafo conexo de G' e $|C^*| = \mathscr{W}(G[L^*])$. Em particular, se L^* induz uma solução ótima para FGkR em G, então $(L^* \cup C^*)$ induz uma solução viável para o novo problema em G', e ambas as soluções têm o mesmo valor.

Reciprocamente, suponha que $G'[L^* \cup C^*]$ seja um subgrafo conexo. Então, o número de componentes de $G[L^*]$ é no máximo $|C^*|$. Em particular, se $(L^* \cup C^*)$ induz uma solução ótima para o novo problema em G' com valor w, então L^* induz uma solução viável para FGkR em G com valor menor ou igual a w.

Concluímos assim que os dois problemas tem soluções ótimas correspondentes e de mesmo valor. □

3.3 A complexidade em função de k

Nesta seção é analisado como o valor de k influencia o problema.

Proposição 3.3.1. *As seguintes afirmações são verdadeiras para um GCA G livre de ciclos monocromáticos e conexo, no contexto do FGkR.*

1. *Para $k = 1$, a solução para o problema é o rótulo que induz o maior número de arestas.*
2. *Existe um valor $k_1 \in \mathbb{Z}$, $1 \leq k_1 \leq |L|$, tal que a solução de FGkR em (G,k_1) é conexa e logo, ótima para FGkR em (G,k) para todo $k \geq k_1$.*
3. *Existem um valor $k_r \in \mathbb{Z}$, $1 \leq \frac{k_r}{k_1} \leq O(1+\ln(|V|-1))$, e uma solução ótima para FGkR em (G,k) conexa, ambos encontrados em tempo polinomial, para todo $k \geq k_r$.*

Demonstração.

1. Como $k = 1$, pela Proposição 3.2.1 basta escolher a floresta geradora monocromática com maior número de arestas. Sendo o grafo livre de ciclos monocromáticos, todo $G[\{l\}]$, $l \in L$, é um grafo acíclico. Logo, a solução é dada pelo rótulo com maior número de arestas.
2. Seja k_1 o valor da solução ótima do MLSTP em G. Por definição, essa é a menor quantidade de cores que induzem uma árvore geradora T de G. Logo, T é a solução do FGkR em (G,k), para todo $k \geq k_1$.
3. É suficiente aplicar o algoritmo MVCA ou rMVCA em G, que retorna o valor k_r e uma árvore geradora com k_r cores, onde k_r tem fator aproximativo em relação à k_1 de $1+\ln(|V|-1)$ (WAN *et al.*, 2002).

□

3.4 Ciclos Coloridos

A partir da Proposição 3.2.1 é possível identificar alguns casos em que FGkR é polinomial. Agora, não somente em função de k, mas também considerando a estrutura do grafo.

Lema 3.4.1. *Seja G um GCA e $k \in \mathbb{Z}^+$. Se todo subgrafo gerador de G com até k cores for acíclico então o FGkR em (G,k) é polinomial.*

Demonstração. Sob as condições do lema, a Proposição 3.2.1 garante que uma solução para FGkR é qualquer subgrafo gerador de G com até k cores e o maior número de arestas. Um tal subgrafo é aquele induzido pelas k cores com maior quantidade de arestas, ou seja, $G[L^*]$ onde $|L^*| = k$ e $|E(c)| \geq |E(l)|$ para todos $c \in L^*$ e $l \in L \setminus L^*$. Um tal L^* pode ser obtido em tempo polinomial, calculando $|E(l)|$ para todo $l \in L$ e escolhendo os k maiores valores. □

Proposição 3.4.2. *Se G for um GCA acíclico então é possível resolver o FGkR em (G,k) em tempo polinomial, para qualquer k.*

Demonstração. Se G for acíclico, também o é todo subgrafo seu. O resultado então segue pelo Lema 3.4.1. □

Corolário 3.4.3. *Se um GCA G for árvore então é possível resolver o FGkR em (G,k) em tempo polinomial, para qualquer k.*

Consideram-se agora GCAs que possuem ciclos grandes. Para isso, define-se abaixo a cintura colorida de um GCA, à semelhança do conceito de cintura de um grafo.

Definição 3.4.4. A cintura de um GCA G, denotada por $g(G)$, é o comprimento do menor ciclo de G. Caso G não possua ciclo considere $g(G)$ como infinito.

Definição 3.4.5. A cintura colorida de um GCA G, denotada por $g'(G)$, é o menor número de cores distintas em um ciclo de G. Caso G não possua ciclo, considere $g'(G)$ como infinito.

Proposição 3.4.6. *Para todo GCA G, $g'(G) \leq g(G)$.*

Demonstração. Suponha por contradição que $g'(G) > g(G)$. Veja que, por definição de $g(G)$, existe um ciclo em G com comprimento $g(G)$. E como esse ciclo possui no máximo uma cor distinta por aresta, ele possui no máximo $g(G)$ cores distintas. Absurdo, dada a hipótese inicial e a Definição 3.4.5. Logo, por redução ao absurdo, tem-se que $g'(G) \leq g(G)$. □

Teorema 3.4.7. *Seja G um GCA. Então o FGkR em (G,k) é polinomial, para todo $k \leq g'(G) - 1$.*

Demonstração. Se $g'(G) \geq k+1$ então, pela Definição 3.4.5, não existe ciclo com k ou menos rótulos distintos. Logo, todo subgrafo induzido por no máximo k rótulos é acíclico, e o resultado segue pelo Lema 3.4.1. □

O teorema acima suscita a questão sobre a complexidade de determinar a cintura colorida de um GCA. O seguinte resultado foi enunciado em (BROERSMA *et al.*, 2005) junto com uma prova com algumas imprecisões, de modo que é apresentada aqui uma prova alternativa.

Teorema 3.4.8. *Encontrar um ciclo colorido mínimo de um GCA G passando por um vértice u é um problema NP-difícil.*

Demonstração. Faz-se a redução a partir do problema de determinar um caminho colorido mínimo em um GCA $G' = (V', E', L', l')$, entre dois vértices não adjacentes s e t, que é sabido ser NP-Difícil (BROERSMA *et al.*, 2005). Construa o grafo G com a adição de um vértice extra u e o faça adjacente a s e t com arestas coloridas com uma nova cor (não pertencente a L'). Então existe um caminho colorido em G' com no máximo r cores se, e somente se, existe um ciclo colorido em G com no máximo $r+1$ cores. Logo, o resultado segue. □

A partir do resultado acima, (BROERSMA *et al.*, 2005) concluem, diretamente, sem qualquer justificativa, que determinar o ciclo com o menor número de cores em um GCA é NP-Difícil. Essa implicação porém não é direta, visto que, em princípio, não é preciso determinar o menor ciclo passando por cada vértice para encontrar o menor de todos. Ao nosso conhecimento, não há uma prova de que, de fato, calcular a cintura colorida seja NP-Difícil.

De todo modo, o interesse sobre essa questão se reduz no contexto do Teorema 3.4.7 pelo fato de que a condição lá requerida para garantir a polinomialidade de FGkR pode ser substituída por outra mais fraca e que pode ser verificada em tempo polinomial.

Teorema 3.4.9. *Seja $G = (V, E, L, l)$ um GCA com $L = \{c_1, \ldots, c_{|L|}\}$ tal que $|E(c_i)| \geq |E(c_{i+1})|$ para $i = 1, \ldots, |L| - 1$. Então o FGkR em (G, k) é polinomial, para todo k tal que $G[c_1, \ldots, c_k]$ seja acíclico.*

Demonstração. Observe que $F^* = G[c_1, \ldots, c_k]$ é um subgrafo gerador com até k cores e maior número possível de arestas. Logo, $|E(F^*)| \geq |E(F)|$ para todo $F \in \mathscr{F}_G(k)$. Além disso, como F^* é uma floresta, F^* é solução ótima pela Proposição 3.2.1. □

4 PROPOSTAS DE SOLUÇÃO

Neste capítulo são propostos modelos matemáticos para o FGkR e também são reportados os desempenhos desses modelos em testes realizados seguindo o padrão estabelecido pela literatura existente para o problema. Veja que o desempenho nos testes será dependente das características específicas de cada modelo, dos métodos de resolução aplicados, do ambiente computacional usado e da implementação que são descritos a seguir.

4.1 Formulações

Apresentamos três modelos de programação inteira para o problema FGkR, a serem referenciados como modelo por floresta, modelo por cortes coloridos e modelo por fluxo. Eles tomam por base as caracterizações do problema derivadas nas proposições 1.2.2, 3.2.1 e 3.2.3. O primeiro modelo é expresso diretamente sobre o grafo G, enquanto os outros dois usam o grafo estendido G'.

As variáveis de decisão utilizadas para definição dos modelos são:

- $x_{uv} \in \{0,1\}$ recebe 1 se $uv \in E(G)$ é usada na solução e, caso contrário, 0.
- $z_l \in \{0,1\}$ recebe 1 se a cor $l \in L \cup C$ está sendo utilizada e, caso contrário, 0.
- $f_{ij} \geq 0$ é quantidade de fluxo de $i \in V(G')$ para $j \in V(G')$.

4.1.1 Modelo por Floresta (FGkR_Floresta)

O modelo por floresta se baseia na caracterização derivada na Proposição 3.2.1. Ele é expresso diretamente sobre o grafo G, e suas soluções viáveis são exatamente florestas de G.

$$\max \quad \sum_{uv \in E(G)} x_{uv} \tag{4.1a}$$

$$\text{s.a.} \quad \sum_{uv \in E(S)} x_{uv} \leq |S| - 1 \quad \forall S \subset V(G), |S| \geq 3, \tag{4.1b}$$

$$\sum_{l \in L} z_l \leq k, \tag{4.1c}$$

$$x_{uv} \leq z_{l(uv)} \qquad \forall uv \in E(G), \tag{4.1d}$$

$$z_l \in \{0,1\} \qquad \forall l \in L, \tag{4.1e}$$

$$x_{uv} \in \{0,1\} \qquad \forall uv \in E(G). \tag{4.1f}$$

A função objetivo (4.1a) maximiza a quantidade de arestas na floresta definida pelas restrições (4.1b) e (4.1f), que são responsáveis pela quebra de ciclos. As desigualdades (4.1c) e limitam o número de rótulos usados, enquanto (4.1d) condicionam a escolha de uma aresta ao uso do seu rótulo.

As restrições (4.1b) são potencialmente em número exponencial, tornando-se impraticável incluí-las todas a priori no modelo. Entretanto, elas podem ser separadas em tempo polinomial (PADBERG; WOLSEY, 1983), mesmo de um ponto fracionário $\bar{x} \in \mathbb{R}^{|E(G)|}$. Em particular, sua separação de um ponto inteiro $\bar{x} \in \mathbb{B}^{|E(G)|}$ consiste na identificação de um ciclo no subgrafo de G induzido pelas arestas $\{uv \in E(G) : x_{uv} = 1\}$, o que pode ser feito por um algoritmo simples de busca, com um DFS.

4.1.2 Modelo por cortes coloridos (FGkR_CCut)

O modelo (FGkR_CCut) segue a formulação CCut proposta por (SILVA *et al.*, 2019) para o MLSTP. Essa formulação aplica a ideia de cortes coloridos (Definição 3.1.5). Para o FGkR, usamos sua equivalência dada na Proposição 3.2.3 e a caracterização de subgrafo conexo expressa na Proposição 3.1.6. Com isso, obtemos o seguinte modelo, expresso em termos do grafo estendido $G' = (V', E', L \cup C, l')$:

$$\min \quad \sum_{l \in C} z_l \tag{4.2a}$$

$$\text{s.a.} \quad \sum_{l \in K(S)} z_l \geq 1 \qquad \forall S \subset V', S \neq \emptyset, \tag{4.2b}$$

$$\sum_{l \in L} z_l \leq k, \tag{4.2c}$$

$$z_l \in \{0,1\} \quad \forall l \in L \cup C. \tag{4.2d}$$

A função objetivo (4.2a) é usada para minimizar o número de cores de C usadas na solução, ou seja, o número de componentes definidas em G. A desigualdade (4.2b) assegura que pelo menos um rótulo de cada corte colorido seja escolhido, garantindo assim que a solução represente um subgrafo conexo. A desigualdade (4.2c) é utilizada para atender o limitante de cores da FGkR. Por último em (4.2d) define-se as variáveis z_l como binárias.

Vale ressaltar que, para esse modelo, quando um rótulo (cor) é usado (valor 1), todas as arestas com aquele rótulo em G' fazem parte da solução do modelo.

Na prática, é inviável aplicar diretamente todas as desigualdades (4.2b). Em nossa implementação, elas são usadas como *Lazy Constraints*. Para isso, é usado um procedimento

que identifica restrições violadas por um ponto inteiro dado, a ser detalhado na Subseção 4.2.2.

4.1.3 Modelo por fluxo (FGkR_Fluxo)

Esse modelo é similar ao de cortes coloridos, com a diferença de que agora a conectividade é garantida enviando um fluxo a partir do vértice universal $s \in V(G')$ para todo outro vértice. Novamente, a Proposição 3.2.3 é base para a corretude do modelo.

$$\min \quad \sum_{l \in C} z_l \tag{4.3a}$$

$$\text{s.a.} \quad f_{sj} + \sum_{i:(i,j)\in E} f_{ij} - \sum_{i:(i,j)\in E} f_{ji} = 1 \qquad \forall j \in V, \tag{4.3b}$$

$$0 \leq f_{ij}, f_{ji} \leq M z_{l(i,j)} \quad \forall (i,j) \in E, \tag{4.3c}$$

$$0 \leq f_{sj} \leq M z_{l(s,j)} \quad \forall j \in V, \tag{4.3d}$$

$$\sum_{l \in L} z_l \leq k, \tag{4.3e}$$

$$z_l \in \{0,1\} \qquad \forall l \in L \cup C. \tag{4.3f}$$

A função objetivo (4.3a) e restrições (4.3e)-(4.3f) são idênticas ao do modelo por cortes coloridos. As igualdades (4.3b) estabelecem um fluxo que inicia no vértice fonte s e é distribuído para todos os vértices de G; logo se um fluxo positivo chega em todo vértice, garante-se que existe um caminho de arestas entre s e todo vértice.

Veja que há duas variáveis, f_{ij} e f_{ji}, para cada aresta (i,j) do grafo original G, que passa a ser representada por dois arcos antiparalelos (um indo de i para j e outro de j para i). Por outro lado, há apenas uma variável, f_{sj}, para cada aresta (s,j) acrescida quando da formação do grafo estendido G'. Note ainda que as restrições (4.3c)-(4.3d) garantem, através de um termo Big-M, que o fluxo só pode atravessar arestas cujos rótulos foram escolhidos. Durante os experimentos foi usado $M = |V|$.

4.1.4 Desigualdades válidas

Apresenta-se aqui uma família de desigualdades válidas para os três modelos propostos acima. Elas limitam o valor da função objetivo em uma solução ótima.

Proposição 4.1.1. *As seguintes desigualdades são válidas para qualquer solução ótima de*

(FGkR_CCut) e (FGkR_Fluxo):

$$\sum_{l\in C} z_l \geq \mathscr{W}(G[\hat{L}]) - \sum_{c\in L\setminus\hat{L}} (\mathscr{W}(G[\hat{L}]) - \mathscr{W}(G[\hat{L}\cup\{c\}]))z_c \quad \forall \hat{L}\subset L. \tag{4.4}$$

E as desigualdades abaixo são válidas para qualquer solução ótima de (FGkR_Floresta):

$$\sum_{uv\in E(G)} x_{uv} \leq |V(G)| - \mathscr{W}(G[\hat{L}]) + \sum_{c\in L\setminus\hat{L}} (\mathscr{W}(G[\hat{L}]) - \mathscr{W}(G[\hat{L}\cup\{c\}]))z_c \quad \forall \hat{L}\subset L. \tag{4.5}$$

Demonstração. Sejam L^* o conjunto de rótulos de uma solução ótima z^* e $\hat{L}\subset L$. Então:

$$\mathscr{W}(G[L^*]) = \mathscr{W}(G[(L^*\cap\hat{L})\cup(L^*\setminus\hat{L})]) \geq \mathscr{W}(G[\hat{L}\cup(L^*\setminus\hat{L})]).$$

E pela Proposição 2.5.1,

$$\mathscr{W}(G[\hat{L}\cup(L^*\setminus\hat{L})]) \geq \mathscr{W}(G[\hat{L}]) - \sum_{c\in L^*\setminus\hat{L}} (\mathscr{W}(G[\hat{L}]) - \mathscr{W}(G[\hat{L}\cup\{c\}])).$$

Então, para o valor da função objetivo em z^* vale:

$$\begin{aligned}\sum_{l\in C} z^*_l = \mathscr{W}(G[L^*]) &\geq \mathscr{W}(G[\hat{L}]) - \sum_{c\in L^*\setminus\hat{L}} (\mathscr{W}(G[\hat{L}]) - \mathscr{W}(G[\hat{L}\cup\{c\}])) \\ &= \mathscr{W}(G[\hat{L}]) - \sum_{c\in L\setminus\hat{L}} (\mathscr{W}(G[\hat{L}]) - \mathscr{W}(G[\hat{L}\cup\{c\}]))z^*_l\end{aligned}$$

Logo a desigualdade (4.4) é válida para z^*. A validade de (4.5) é demonstrada de forma similar, bastando observar a relação $\sum_{l\in C} z_l = |V(G)| - \sum_{uv} x_{uv}$ entre as funções objetivo de (FGkR_CCut) e (FGkR_Floresta). □

As desigualdades acima podem ser expressas de outra forma. Por exemplo, dado um limite superior UB para (FGkR_CCut), dado por uma solução viável conhecida, pode-se considerar a desigualdade $\sum_{l\in C} z_l \leq UB-1$ para restringir a busca por soluções melhores. Junto com (4.4), essa desigualdade leva a:

$$\sum_{c\in L\setminus\hat{L}} (\mathscr{W}(G[\hat{L}]) - \mathscr{W}(G[\hat{L}\cup\{c\}]))z_c \geq \mathscr{W}(G[\hat{L}]) - UB + 1 \quad \forall \hat{L}\subset L. \tag{4.6}$$

4.2 Resolução dos modelos

Nesta seção são sugeridas ideias e o que é necessário para resolver os modelos. Em certa medida, todas elas foram avaliadas computacionalmente, mas apenas algumas são efetivamente usadas nos experimentos reportados neste capítulo.

4.2.1 Modelos Iniciais

Os modelos iniciais passados para o *solver Cplex* não são exatamente aqueles descritos na seção anterior.

No caso de (FGkR_Floresta), as restrições (4.1b), que são em número exponencial, serão omitidas do modelo inicial e adicionadas iterativamente, através de um processo de geração de restrições (*lazy constraints*), descrito na próxima subseção.

De modo parecido, todas ou quase todas as restrições (4.2b) serão omitidas do modelo (FGkR_CCut) inicial e incluídas a posteriori por um processo de geração de restrições, quando violadas. Por exemplo, (SILVA, 2018) sugere utilizar apenas as restrições (4.2b) definidas por conjuntos unitários, isto é,

$$\sum_{l \in K(\{v\})} z_l \geq 1, \forall v \in V'. \tag{4.7}$$

Por outro lado, o modelo (FGkR_CCut) inicial pode conter a seguinte desigualdade sugerida por (SILVA, 2018) para o MLSTP:

$$\sum_{l \in L \cup C} |E'(\{l\})| \cdot z_l \geq |V'| - 1. \tag{4.8}$$

Como o número de arestas em qualquer subgrafo gerador conexo de G' tem pelo menos $|V'| - 1$ arestas e cada rótulo $l \in L \cup C$ escolhido contribui com $|E'(\{l\})|$ dessas arestas, a desigualdade (4.8) é válida para uma solução do FGkR no grafo estendido. As restrições (4.7)-(4.8) também são válidas no modelo inicial (FGkR_Fluxo).

Ambas as desigualdades possuem certo impacto para diferentes instâncias. Quando um grafo tem vértices com poucas cores incidentes a ele, a Equação 4.7 é interessante. Já em grafos com poucas arestas, a Equação 4.8 pode remover uma grande quantidade soluções fracionárias. (SILVA, 2018) utiliza por preferência as restrições do tipo (4.7). Entretanto, como as desigualdades de ambos os tipos são válidas, é possível utilizar um subconjunto qualquer delas no modelo inicial.

Além dessas modificações, os modelos (FGkR_CCut) e (FGkR_Fluxo) iniciais forçam as soluções a terem exatamente k cores, ou seja, considera-se igualdade em (4.2c) e (4.3e). Perceba que não se pode fazer o mesmo com o modelo (FGkR_Floresta), pois uma solução (acíclica) ótima pode, de fato, usar menos que k cores.

4.2.2 Lazy Constraints

As restrições (4.2b) e (4.1b) são acrescidas aos respectivos modelos por um procedimento *LazyConstraintCallback* do CPLEX. Ele verifica se a solução 0-1, ou seja, inteira obtida induz um subgrafo conexo do grafo estendido, no caso de (FGkR_CCut), ou um subgrafo acíclico (floresta geradora) do grafo original, no caso de (FGkR_Floresta).

Para isso, as variáveis z com valor 1 na solução são determinadas e são codificadas em um *bitmap*. A partir desse *bitmap*, usa-se uma interface que reduz o grafo base do modelo, mantendo somente as arestas cujas cores estão ativas (variáveis z com valor 1), ou seja, obtém-se o subgrafo induzido pelas cores ativas. Por fim, aplica-se uma DFS, que classifica os vértices por componente e retorna o número de componentes do grafo reduzido. Essa operação se repete algumas vezes neste trabalho, pois a partir da classificação dos vértices em componentes se estudam relações com as arestas do grafo original.

No caso do modelo (FGkR_CCut), é verificado se o número de componentes é igual a 1, ou seja, se o grafo reduzido é conexo. Caso seja conexo, temos uma solução inteira válida para o (FGkR_CCut). Do contrário, o grafo reduzido não é conexo e alguma restrição de corte colorido está sendo violada. Cada componente W_i, $i = 0,1,2,...,p$, desse grafo define um corte colorido $K(W_i) = \{l(e) : e = (u,v) \in E', u \in W_i, v \notin W_i\}$ da seguinte forma:

$$\sum_{l \in K(W_i)} z_l \geq 1, \quad i = 0,1,2,...,p. \tag{4.9}$$

Adicionando esse conjunto de restrições, o problema relaxado seria fortalecido, invalidando a solução inteira passada para o *LazyConstraintCallback*. Veja porém que existe a possibilidade de se identificar vários outros cortes coloridos violados, "agrupando" componentes desse grafo com a "ativação" de outras cores. Mais precisamente, se L_z é o conjunto de cores ativas na solução z, $\ell \in (L \cup C) \setminus L_z$ e $G'[L_z \cup \{\ell\}]$ é desconexo, pode-se gerar uma desigualdade violada da forma (4.9) para cada uma das p_ℓ componentes de $G'[L_z \cup \{\ell\}]$. Com isso, o número de desigualdades geradas passa a ser no máximo igual ao anterior, pois $p_\ell \leq p$, e o número de variáveis z nelas envolvidas torna-se estritamente menor, já que z_ℓ não aparece mais. Isto vai ao encontro do fato de que, quanto menor o número de cores que definem o lado esquerdo da desigualdade (4.9), mais forte ela se torna.

Nessa perspectiva, o seguinte procedimento de perturbação é proposto. Iterativamente, procura-se ativar mais cores no grafo reduzido de forma que ele ainda continue desconexo. Em outras palavras, procura-se uma cor ainda não presente nesse grafo, tal que ele se mantenha

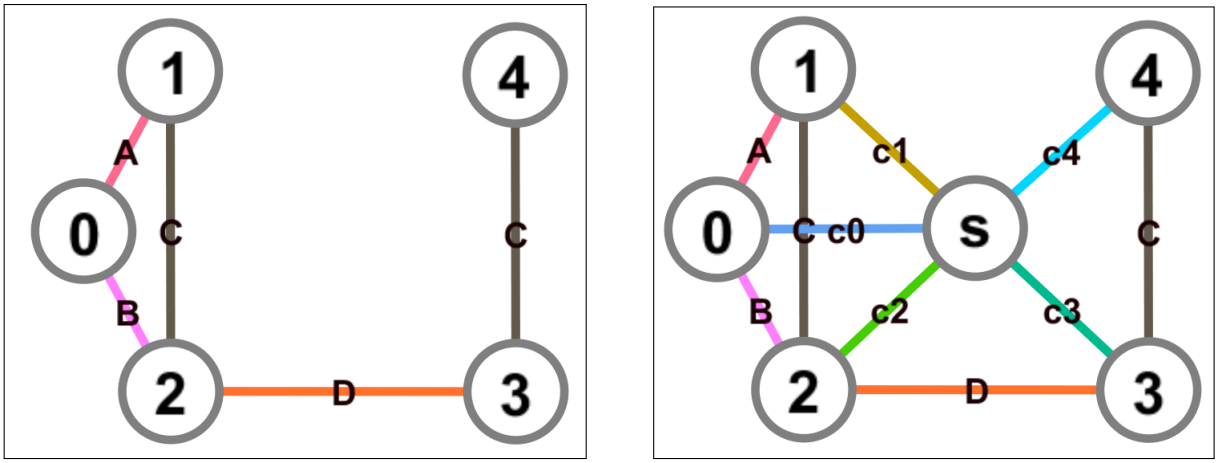
desconexo, mesmo com a introdução de todas as arestas dessa cor. Esse procedimento pode ser realizado, por exemplo, por uma heurística gulosa baseada no kMVCA, onde cada passo tem como objetivo encontrar uma cor que, quando ativa, diminui o número de componentes do menor valor possível. Com essa estratégia gulosa, procura-se ativar o maior número de cores antes de o grafo se tornar conexo. Vale ressalvar que, para implementação desse procedimento de perturbação, uma estrutura de conjuntos disjuntos é mais interessante, em termos de complexidade assintótica, que a aplicação de DFS, assim como acontece na implementação do rMVCA (SILVA, 2018).

Observe que, a cada iteração desse processo de perturbação, pode-se gerar um conjunto de desigualdades de cortes coloridos violadas, possivelmente diferente daquele da iteração anterior, mas com alguma interseção com este. O número de desigualdades geradas vai decrescendo a cada iteração e pode chegar a ser unitário, quando o grafo a ela associado tiver apenas 2 componentes.

Para ilustrar a aplicação do processo de perturbação, considere o exemplo a seguir. A Figura 7 apresenta o grafo original e estendido, respectivamente. As demais figuras ilustram a separação da solução definida por $z_A = z_{c_4} = 1$ e as demais variáveis z iguais a 0. Na Figura 8 observa-se o grafo reduzido e as desigualdades do tipo (4.9) geradas. Com a ativação da cor B, obtêm-se o grafo reduzido e um conjunto de três desigualdades violadas apresentados na Figura 9. Posteriormente, com a ativação da cor C, gera-se o grafo reduzido e uma restrição, exibidos na Figura 10. Nesse ponto, nenhuma outra cor pode ser ativada sem tornar o grafo conexo.

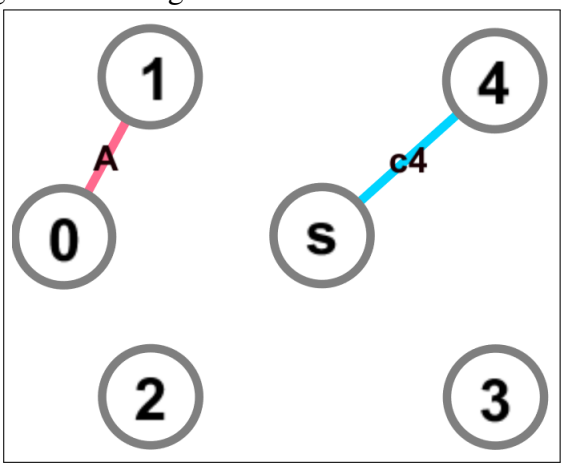
Em lugar de adicionar ao modelo apenas os cortes (4.9) definidos pelas componentes

Figura 7 – Grafo exemplo original e estendido



Fonte: elaborado pelo autor (2020).

Figura 8 – Subgrafo reduzido

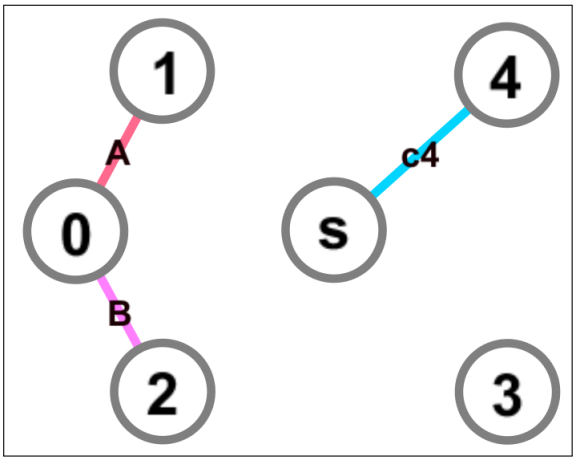


Cortes coloridos violados

$$z_{c0} + z_{c1} + z_C + z_D \geq 1$$
$$z_{c2} + z_B + z_C + z_D \geq 1$$
$$z_{c3} + z_C + z_D \geq 1$$
$$z_{c0} + z_{c1} + z_{c2} + z_{c3} + z_C \geq 1$$

Fonte: elaborado pelo autor (2020).

Figura 9 – Subgrafo reduzido com ativação da cor B

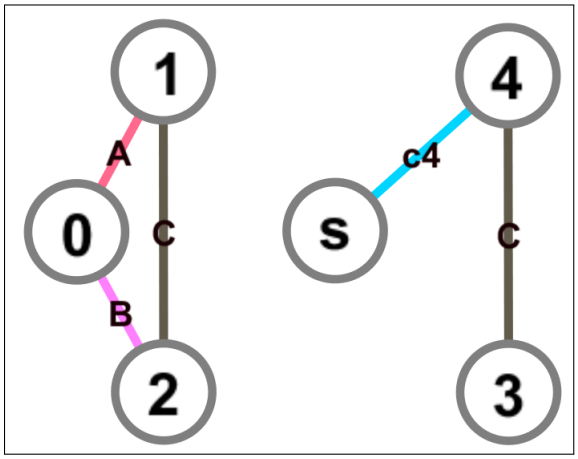


Cortes coloridos violados

$$z_{c0} + z_{c1} + z_{c2} + z_D \geq 1$$
$$z_{c3} + z_C + z_D \geq 1$$
$$z_{c0} + z_{c1} + z_{c2} + z_{c3} + z_C \geq 1$$

Fonte: elaborado pelo autor (2020).

Figura 10 – Subgrafo reduzido com ativação das cores B e C



Corte colorido violado

$$z_{c0} + z_{c1} + z_{c2} + z_D \geq 1$$

Fonte: elaborado pelo autor (2020).

do grafo reduzido inicial, pode-se fazer isso com respeito às componentes de cada novo grafo criado pelo processo de perturbação, enquanto ele continuar desconexo. Com isso, pode-se obter uma maior família de cortes violados. Entretanto, é preciso ter controle sobre os cortes repetidos

de uma iteração para outra ou mesmo sobre o número total de desigualdades adicionadas.

Nos experimentos apresentados neste capítulo, optou-se por duas alternativas. Na primeira, são analisadas apenas as desigualdades (4.9) relativas às componentes do grafo reduzido inicial. Na segunda, são considerados apenas os cortes gerados na última iteração do processo de perturbação, de modo a minimizar a inclusão de cortes repetidos e a manter um número baixo de desigualdades no modelo. Além disso, após testes preliminares, indica-se que sejam adicionados ao modelo do *Cplex* como cortes locais, de modo a serem considerados apenas para o nó corrente e seus descendentes na árvore de *Branch&Cut*.

Para o modelo (FG*k*R_Floresta), uma solução inteira encontrada informa quais as arestas usadas. Caso a solução seja válida, ou seja, defina uma floresta, o grafo reduzido deve possuir o número de componentes igual a diferença de $V(G)$ e o número de arestas ativas, conforme Proposição 2.1.3. O *LazyConstraintCallback* usado faz essa comparação e, caso falhe, o grafo reduzido contém ciclos. Nesse caso, desigualdades do tipo (4.1b) são adicionadas para invalidar a solução atual. Sabe-se que a DFS classifica os vértices em componentes W_i. Para quebrar os ciclos, define-se $H_i = \{uv \in E(G) : u, v \in W_i\}$ e adiciona-se o seguinte corte de forma local, caso esteja violado:

$$\sum_{uv \in H_i} x_{uv} \leq |W_i| - 1, \qquad i = 0, 1, 2, ..., p \tag{4.10}$$

Com esse métodos, os problemas de separação das desigualdades (4.2b) e (4.1b) são resolvidos. Vale ressaltar que o procedimento de separação para o modelo (FG*k*R_Floresta) funciona apenas porque a solução a ser separada é inteira. Para soluções fracionárias existe um procedimento de separação polinomial em (PADBERG; WOLSEY, 1983).

4.2.3 Planos de Cortes

Como comentado, as Equações (4.9) vão sendo iterativamente adicionadas ao modelo relaxado inicial de (FG*k*R_CCut) e, de uma certa forma, ajudam a selecionar quais cores são necessárias para uma solução conexa. Por exemplo, se um corte colorido é monocromático, conclui-se que esta cor faz parte da solução ótima, para esse modelo. Com essa intuição, um procedimento de planos de corte usando as Equações (4.9) como cortes é desenvolvido.

Para tal, usa-se o método *UserCutCallback* provido pelo *Cplex*. Diferentemente dos procedimentos do tipo *LazyConstraint* apresentados na subseção anterior, aplicados quando se encontra uma solução inteira, agora são usadas as soluções fracionárias das relaxações lineares

dos nós do *Branch&Cut* processados. Dada uma tal solução, os valores das variáveis são "arredondados" para 1, caso maiores que zero (a depender do nível de precisão usado pelo comparador '$>$' na linguagem c++), ou para 0, caso contrário. Cria-se assim uma solução inteira. Perceba, porém, que o grafo reduzido por ela gerado não necessariamente possui k rótulos. De qualquer modo, a partir dessa solução, aplica-se o mesmo procedimento que foi usado para o *LazyConstraintCallback* de (FGkR_CCut) com uma diferença: os cortes adicionados serão globais, valendo a partir daí durante todo o processo enumerativo.

4.2.4 Limites

Dentre os fatores que afetam um algoritmo de *Branch&Cut*, os limites inferiores e superiores são muito importantes. Aqui, o valor objetivo da solução da heurística gulosa encontrada usando kMVCA pode ser adaptado para os três modelos como limite superior. Esse limite permite os métodos convergirem mais rápido para uma solução tão boa quanto a heurística usada. Como limite inferior, usa-se o valor trivial (ou seja, 1) referente a um grafo conexo (uma componente).

4.2.5 Poda e Ramificação

Assim como no algoritmo de *backtracking*, apresentado na Subseção 2.5.2, a Proposição 2.5.1 fornece um critério de poda. No nó corrente da árvore de *Branch&Cut*, pode-se verificar as variáveis z já fixadas em 0 ou em 1 via o método *BranchCallback*, disponibilizado pelo *CPLEX*. Essas variáveis fixadas definem uma solução parcial, como no algoritmo de *backtracking*. Sendo assim, pode-se estimar a maior redução possível no número de componentes dessa solução parcial, e podar o nó caso não seja possível gerar solução melhor que uma já conhecida.

Veja que cada *Callback* incorporado ao método de resolução do modelo é anexado de alguma forma ao processamento dos nós, aumentando o tempo para que cada nó seja processado. Dessa forma, deve-se ter cuidado com as chamadas a esses procedimentos, especialmente quando se pode presumir que sua aplicação não sortirá o efeito desejado (um bom corte, uma poda etc). Por exemplo, a poda por limite inferior dada pela Proposição 2.5.1 é implementada por um *BranchCallback*. Porém, só vale a pena efetuar o cálculo de $maxReduction$ quando a solução parcial já possuir um número de variáveis fixas em 1's grande o suficiente para evitar o processamento adicional excessivo sem a expectativa de poda. Em particular, não se aplica o

teste de poda até que a busca (*Branch&Cut*) encontre uma solução inteira válida inicial. De todo modo, não é certo que o custo da utilização de um *BranchCallback* específico para cálculo do limite inferior compense uma eventual poda, sendo necessários experimentos computacionais para essa verificação.

Dada a importância da seleção da variável fracionária a ser ramificada no método *Branch&Cut*, uma sequência de prioridade das variáveis pode ser informada ao *Cplex*. As diferentes formas de organizar essa sequência afetam o desempenho do método. No sentido de se aproximar das caraterísticas do Algoritmo Enumerativo, sugere-se, a princípio, dar preferência à ramificação em variáveis que representem cores com maior número de arestas.

4.2.6 Pré-processamento e fixação de variáveis

As soluções parciais ensejam a aplicação de outros elementos aos métodos de solução, como a noção de dominância (Subseção 3.1.3). Lembre que uma cor c é dominada em uma solução parcial que não contém c se as arestas de cor c são todas internas às componentes do subgrafo definido por essa solução parcial, ou seja, as extremidades de cada aresta colorida com c pertencem a uma mesma componente desse grafo. Conclui-se que essa cor, ao ser adicionada à solução parcial, não irá diminuir o número de componentes. Então, variáveis correspondentes a cores dominadas podem ser fixadas em zero. Porém, veja que o modelo usado não poderia fixar o número de cores em exatamente k, caso essa fixação fosse implementada. A efetividade dessa estratégia de fixação de variáveis depende bastante da estrutura das instâncias usadas nos testes, sendo mais promissora naquelas onde muitas cores são dominadas por um subconjunto relativamente pequeno de outras cores. Espera-se que em grafos com grande densidade seja possível encontrar esse tipo de instâncias.

Antes de processar o grafo de entrada, é possível aplicar um pré-processamento para quebra de ciclos monocromáticos, ou seja, ciclos definidos por arestas com o mesmo rótulo. Como mostra a Proposição 3.1.3, a quebra desses ciclos gera uma instância equivalente, com mesma solução ótima. De fato, a remoção de uma aresta que está em um ciclo monocromático não altera o número de cores nem o número de componentes da solução. Vale mencionar que esse pré-processamento pode ser feito em todas as formulações.

Na verdade, como discutido na subseção 3.1.1, dada uma componente monocromática no grafo de entrada, pode-se modificar suas arestas de forma a obter qualquer subgrafo conexo monocromático (na mesma cor) envolvendo os vértices da componente, desde um grafo

completo a uma árvore, sem mudar o problema. Entretanto, tais transformações levam à necessidade de adaptar as formulações para multigrafos. Testes iniciais não mostraram resultados promissores com tais transformações, principalmente para o modelo de fluxo. Encontrar uma forma de aplicar transformações para obter modelos mais fortes não é trivial. Por esse motivo, elas não serão mostradas neste trabalho, apesar de terem sido testadas nos experimentos iniciais.

4.3 Resultados preliminares

Nesta seção são definidos o ambiente de testes, como as instâncias são geradas seguindo a literatura e os métodos usados. Inicialmente, são feitos testes usando configurações mais simples para obter um melhor contraste com os ajustes propostos.

4.3.1 Ambiente Computacional dos testes iniciais

Testes iniciais foram executados usando máquina com processador i7-3770 3.40Ghz no modo *Turbo Boost*, 8GB de memória RAM DDR3 e disco rígido de 1TB. Implementações foram feitas em C++ usando Cplex 12.8 e as bibliotecas providas pela *Boost C++ Libraries*. O algoritmo enumerativo faz um *backtracking* recursivo usando 1 *thread*, e as outras implementações usam 6 *threads*, pois foi mais estável que 8 para configuração de hardware usada, em modo determinístico. Vale ressaltar que, por padrão, o *Cplex* desativa multiprocessamento quando é usado *Control Callbacks* (*LazyContraintCallback* e *UserCutCallback*), porém esse pode ser forçado quando sua implementação é segura em *threads*. Os testes reportados usam prioridade para viabilidade das soluções, pois produz melhores resultados no modelos.

4.3.2 Banco de instâncias de testes

Os testes são baseados em (CERULLI *et al.*, 2014), que sugerem o uso de instâncias obtidas da seguinte maneira. Cada instância, um grafo $G = (V, E, L, l)$ rotulado em arestas e um inteiro k, é identificada por $n(G)$-$m(G)$-$|L|$-k-id e gerada de forma pseudo-aleatória, usando o gerador *Mersenne twister* (na biblioteca *boost* como mt19937) e aplicando uma distribuição uniforme para geração de arestas e rótulos, garantindo que todo rótulo possui pelo menos uma aresta associada. Cada grafo é gerado com os seguintes parâmetros: $n(G) \in \{100, 200, 300, 400, 500, 1000\}$, $m(G) = \frac{n(G)\cdot(n(G)-1)}{10}$, $|L| \in \{\lfloor n(G)/4 \rfloor, n(G)/2, n, \lfloor 1.25 \cdot n(G) \rfloor\}$ e um identificador de instância $id \in \{1, 2, ..., 10\}$. Assim, tem-se no total 240 instâncias.

(CERULLI *et al.*, 2014) propõem encontrar o k de forma empírica, tal que o problema não seja tão fácil de resolver. Para tanto, usam $k = \frac{n(G)}{2^i}$, onde i é o menor valor para o qual o kMVCA não reporta uma solução trivial (com uma única componente).

4.3.3 Experimentos Iniciais

Aqui são apresentados resultados de testes iniciais realizados com o algoritmo enumerativo, o *Branch&Cut* padrão do CPLEX aplicado a FGkR_Fluxo, e uma versão usando *Callbacks* aplicada a FGkR_CCut. As comparações são feitas por meio de gráficos, construídos com base nos dados detalhados que se encontram na Tabela 3 do Apêndice A.

O tempo limite para resolução de cada instância foi estipulado em 2 horas (7200s) para cada método. Observa-se que o tempo total reportado nas tabelas e gráficos pode exceder um pouco esse limite devido a processamento necessário à resolução, como alguns cortes gerados pelo *Cplex* para formulação de fluxo no começo do processo de *Branch&Cut*.

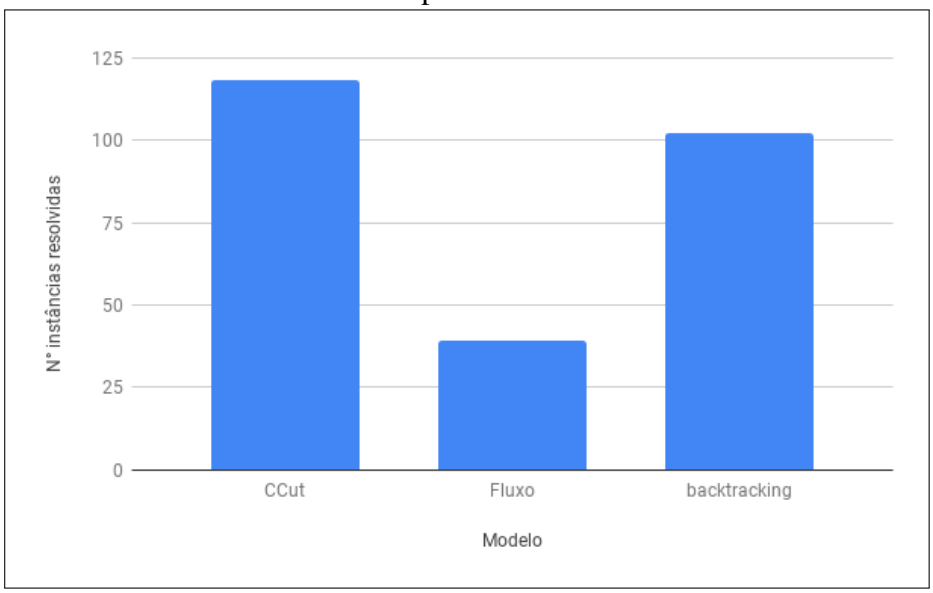
Nos primeiros experimentos com o *Branch&Cut* do *Cplex* procurou-se avaliar o desempenho do método junto aos modelos, sem inferência de estratégias específicas. Por esse motivo, o primeiro *Branch&Cut* testado não usa alguns dos elementos propostos na seção anterior. O modelo FGkR_Fluxo, dado por (4.3a)-(4.3f), foi diretamente submetido ao *Cplex*, para aplicação do seu *Branch&Cut*. Já o FGkR_CCut foi utilizado da seguinte forma:

- Para fins de fortalecimento do modelo (FGkR_CCut) inicial, foi utilizada a restrição (4.8). Essa obteve um resultado melhor em testes que as desigualdades (4.7) para os grafos usados nos testes preliminares.
- As desigualdades de cortes coloridos (4.2b) são separadas via *LazyConstraintCallback* (subseção 4.2.2) porém sem o processo de pertubação e usando somente uma desigualdade válida violada referente a uma componente fixa arbitrária (no caso a componente com menor índice).
- A adição de planos de corte, via *UserConstraintCallback*, segue processo de geração de desigualdades violadas que foi estabelecido no item anterior.

Limites, podas e as regras de ramificação são aquelas existentes no *Cplex*. Em todos os testes feitos no *Cplex*, os parâmetros de *presolve* foram desativados. O caractere '-' em uma entrada da Tabela representa o valor máximo para aquele campo (7200s caso enumerativo).

O gráfico da Figura 11 é baseado nesses testes e mostra um comparativo da quantidade de instâncias resolvidas por cada método. É possível ver claramente que o modelo baseado

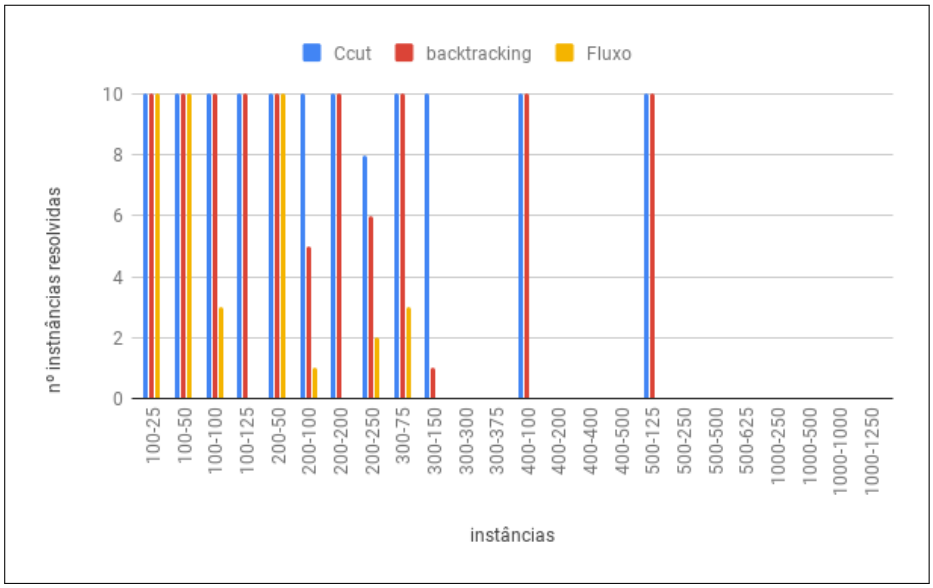
Figura 11 – Total de instâncias resolvidas por método



Fonte: elaborado pelo autor (2020).

em fluxo possui um desempenho baixo. Por outro lado, das 240 instâncias, o modelo FGkR_CCut resolve 118 instâncias e o algoritmo enumerativo, 102. Para uma comparação mais detalhada, veja os gráficos das Figuras 12 e 13, que exibem, respectivamente, o número de instâncias resolvidas e os tempos médios por grupos de instâncias, separadas pelo quantidade de vértices e cores.

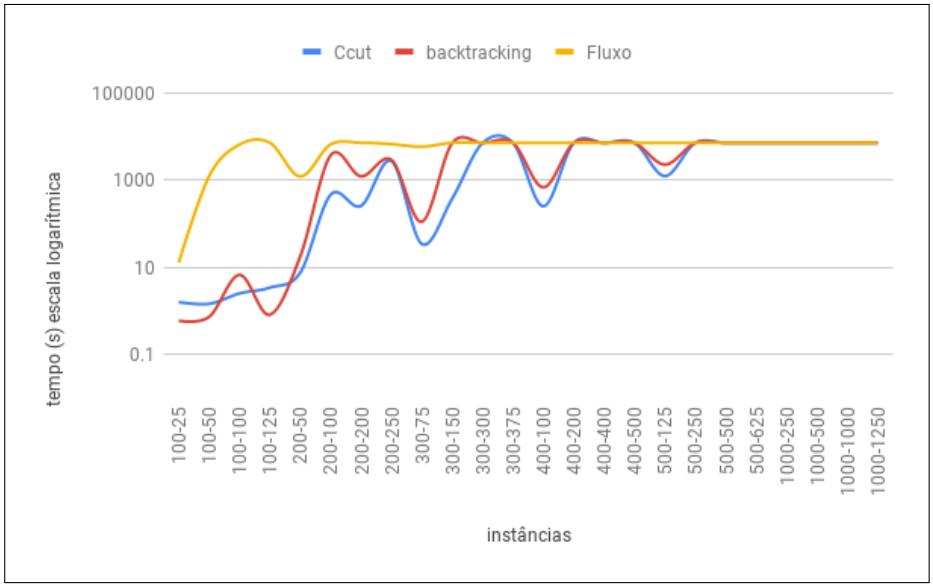
Figura 12 – Total de instâncias – separadas por (número de vértices)-(número de rótulos) – resolvidas por método



Fonte: elaborado pelo autor (2020).

Aponta-se aqui a omissão dos testes com modelo por floresta, dado que esse apresentou resultados muito pouco satisfatórios, até mesmo para instâncias pequenas. Ressalve-se,

Figura 13 – Tempo médio de resolução das instâncias – separadas por (número de vértices)-(número de rótulos) – por modelo



Fonte: elaborado pelo autor (2020).

porém, que foi aplicado o procedimento de separação (somente para pontos inteiros), usando DFS, conforme *LazyConstraintCallback* proposto na seção anterior. Vale mencionar ainda que, alternativamente às restrições (4.1b), foram testadas restrições MTZ para eliminação de ciclos (MILLER *et al.*, 1960). Os testes preliminares não foram promissores.

4.4 Aplicando melhorias

Esta seção tem como objetivo mostrar um conjunto de ajustes feitos aos modelos com escopo relativo aos testes usados.

4.4.1 Paralelizando o Algoritmo Enumerativo

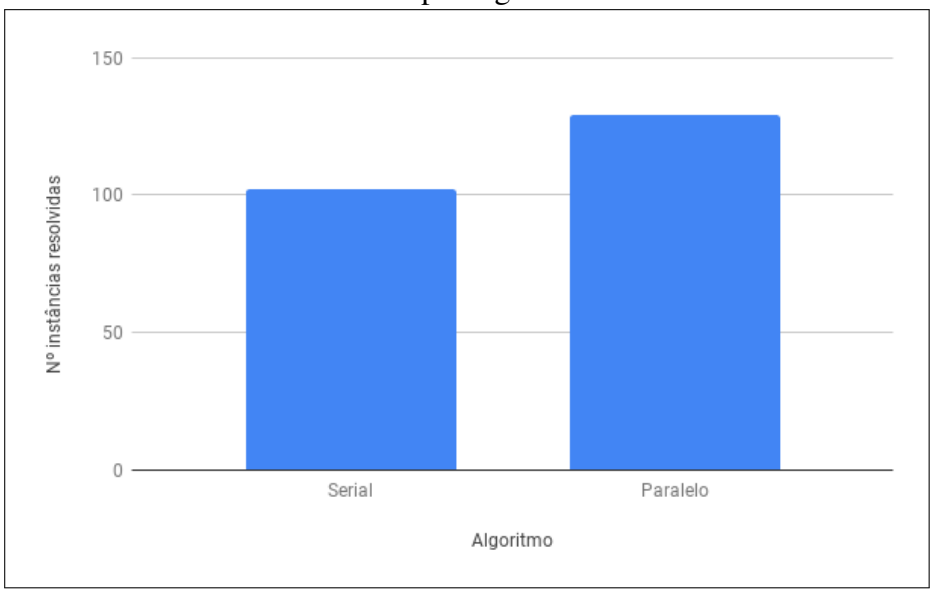
A paralelização do Algoritmo 5 é feita usando C++ com a biblioteca *Threading Building Blocks* (TBB) da Intel, onde o nó inicial é alocado em uma lista inicial de tarefas e passado para o método genérico da biblioteca *parallel_do*. Esse método irá processar cada nó da lista, alocando apropriadamente recursos e sincronizando *threads*. O algoritmo paralelo utiliza de pré-processamento para quebra de ciclos monocromáticos, posto que sua aplicação melhora o desempenho, diferentemente do que acontece com os modelos.

Cada nó é processado por um método que funciona de forma similar ao Algoritmo 5, porém, para evitar comunicação entre os nós, a melhor solução encontrada é mantida em uma variável global atômica, acessada por um método não-bloqueante *compare_and_swap*, que

verifica de forma atômica se o valor da variável é consistente, antes de mudar seu valor; caso contrário, o fluxo lógico é reiniciado, na forma de um laço, para um valor atualizado. O processo de ramificação é estabelecido pela criação de dois nós, que alimentam o método *parallel_feeder*, responsável por alocar os nós na fila de tarefas. O algoritmo termina quando todos os elementos da fila de tarefas são processados.

Os testes com a implementação paralela do algoritmo enumerativo podem ser encontrados no Apêndice A, na Tabela 4. Na Figura 14 vê-se que a paralelização permitiu o algoritmo enumerativo resolver mais instâncias. Adicionalmente, é possível ver na Figura 15 que instâncias grandes de 500 e 1000 vértices são resolvidas, o que mostra que o método resolve mais problemas que o *Branch&Cut* aplicado ao FG*k*R na subseção 4.3.3. Na Figura 16 é apresentado um comparativo de tempo médio para instâncias com o eixo *y* em escala logarítmica.

Figura 14 – Total de instâncias resolvidas por algoritmo



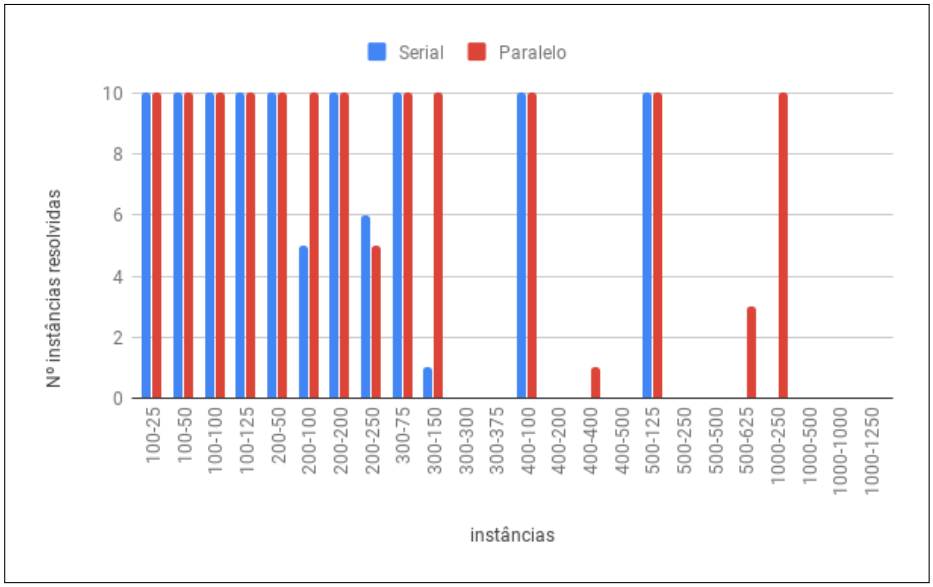
Fonte: elaborado pelo autor (2020).

Veja que o algoritmo paralelo não processa os nós na mesma ordem que o algoritmo serial, principalmente por que os nós processados em paralelo são escalonados através de uma estrutura de fila. Isso faz com que algumas instâncias tenham sido resolvidas pelo serial e não pelo paralelo, dentro do tempo limite.

4.4.2 Ajustando o modelo FGkR_CCut

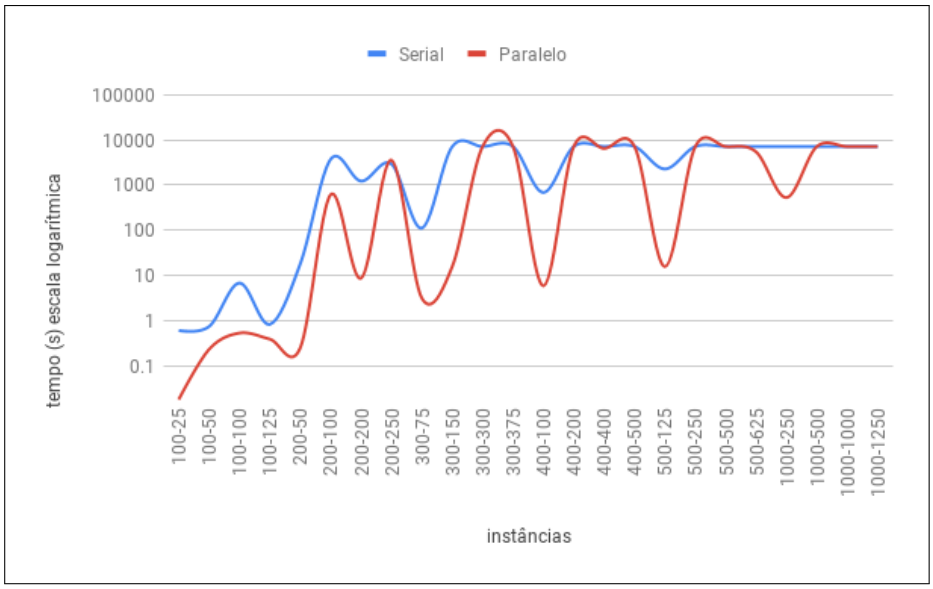
Dada a melhoria de desempenho do algoritmo enumerativo, em sua implementação paralela, conforme Subseção 4.4.1, deseja-se comparar os novos resultados com o modelo

Figura 15 – Total de instâncias – separadas por (número de vértices)-(número de rótulos) – resolvidas por algoritmo



Fonte: elaborado pelo autor (2020).

Figura 16 – Tempo médio de resolução das instâncias – separadas por (número de vértices)-(número de rótulos) – resolvidas por algoritmo



Fonte: elaborado pelo autor (2020).

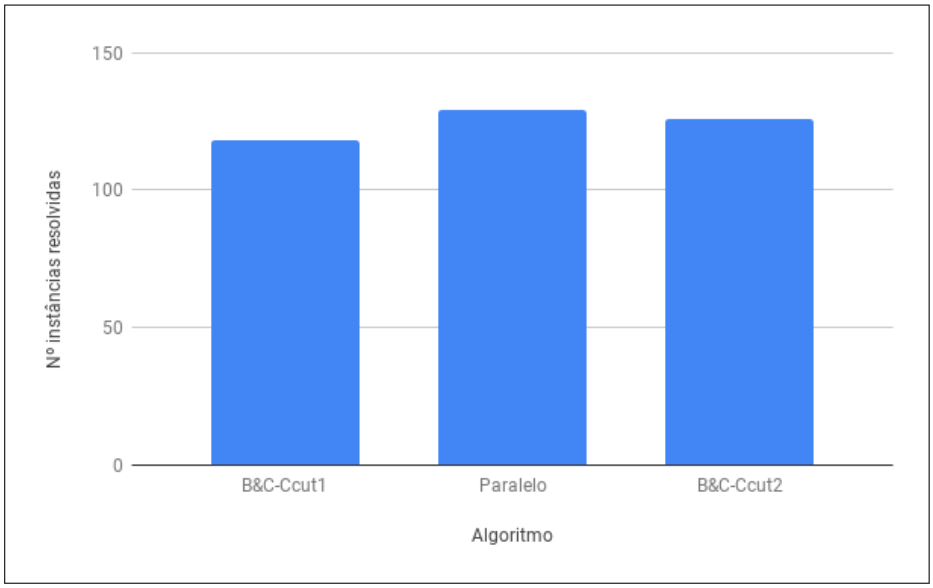
FG*k*R_CCut. Agora, serão também incorporadas ao *Branch&Cut* algumas das ideias que foram propostas anteriormente, na Seção 4.2. Essa nova versão, melhorada, será referenciada como *B&C-Ccut2*, enquanto a versão anterior, que nos gráficos anteriores é chamada simplesmente de *Ccut*, passa a ser *B&C-Ccut1*. Veja no Apêndice A uma tabela mais detalhada (Tabela 5).

A versão *B&C-Ccut2* é executada em um ambiente computacional parecido com o anterior, porém agora usando 8 *threads* em modo determinístico e 16GB de memória DDR3. Além disso, tanto o *CutCallback* quanto o *LazyConstraintCallback* são feitos como explicado na

seção 4.2, as variáveis de cores são ramificadas por ordem decrescente de número de arestas, o *k*MVCA é usado para encontrar o limite superior e é inserido o limite inferior trivial. Vale ressaltar que, pela natureza dos cortes usados, o *Cplex* é configurado para priorizar a viabilidade das soluções. Todos os outros procedimentos são idênticos àqueles usados no *B&C-Ccut1*. Em particular, pela estrutura dos grafos utilizados, optou-se por não aplicar o *BranchCallback* para poda por limite inferior e dominância. Testes mostram que o impacto desses procedimentos para essas instâncias é mínimo.

Como feito anteriormente, os resultados detalhados estão anexados no Apêndice A. A Figura 17 mostra que as melhorias permitiram o modelo resolver mais instâncias (126), porém ainda menos instâncias que o algoritmo enumerativo paralelo (129). É possível ver na Figura 18 uma certa robustez para o novo método de resolução em instâncias com 100-200 vértices ou problemas com até 150 rótulos. Com as modificações, os tempos de resolução (ver Figura 19) se aproximaram muito do algoritmo enumerativo paralelo, mas esse ainda continua sendo mais rápido.

Figura 17 – Número de instâncias resolvidas comparativo com *Branch&Cut* melhorado

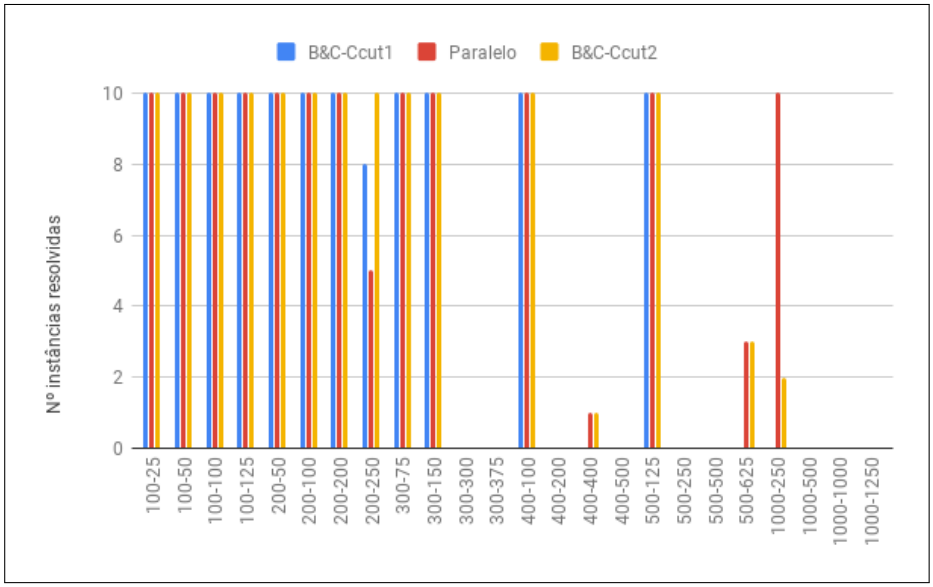


Fonte: elaborado pelo autor (2020).

4.4.3 Decompondo o modelo FGkR_Fluxo via Benders

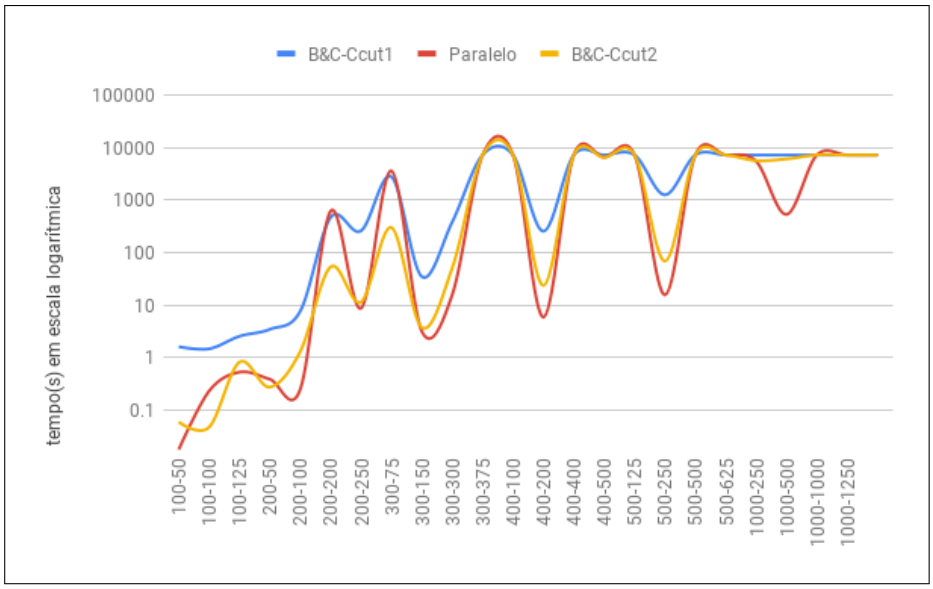
Sabe-se que, diferentemente do modelo por cortes coloridos, o modelos de fluxo começa com uma relaxação linear mais forte, dado que possui um número de restrições polinomial. Por outro lado, é visto na subseção 4.3.3 que os resultados não confirmam o melhor desempenho

Figura 18 – Número de instâncias resolvidas comparativo com *Branch&Cut* melhorado separado



Fonte: elaborado pelo autor (2020).

Figura 19 – Tempo médio de resolução das instâncias comparativo com *Branch&Cut* melhorado separado



Fonte: elaborado pelo autor (2020).

que seria esperado para a formulação, o que pode ter sido consequência do grande número de variáveis continuas de fluxo. Com o intuito de incrementar esse desempenho, nesta subseção propõe-se a utilização da decomposição de Benders (seção 2.4). Serão derivados os problemas mestre e escravo, bem como reportados testes comparativos com a estratégia de resolução usada na seção anterior.

Para tanto, seja (P) o modelo (FGkR_Fluxo):

$$(P)\ \min \quad \sum_{l\in C} z_l \tag{4.11a}$$

$$\text{s.a.} \quad f_{sj} + \sum_{i:(i,j)\in E} f_{ij} - \sum_{i:(i,j)\in E} f_{ji} = 1 \qquad \forall j \in V, \tag{4.11b}$$

$$M z_{l(i,j)} - f_{ij} \geq 0 \qquad \forall (i,j) \in E, \tag{4.11c}$$

$$M z_{l(i,j)} - f_{ji} \geq 0 \qquad \forall (i,j) \in E, \tag{4.11d}$$

$$M z_{l(s,j)} - f_{sj} \geq 0 \qquad \forall j \in V, \tag{4.11e}$$

$$\sum_{l\in L} z_l \leq k, \tag{4.11f}$$

$$z_l \in \{0,1\} \quad \forall l \in L \cup C, \tag{4.11g}$$

$$f_{ij}, f_{ji} \geq 0 \qquad \forall ij \in E, \tag{4.11h}$$

$$f_{sj} \geq 0 \qquad \forall j \in V. \tag{4.11i}$$

Expressando as variáveis contínuas em função das inteiras, obtém-se o seguinte modelo, onde $\bar{z} \in \{\{0,1\}^{|L\cup C|} : \sum_{l\in L} \bar{z}_l \leq k\}$:

$$P(\bar{z})\ \min \quad 0^T f \tag{4.12a}$$

$$\text{s.a.} \quad f_{sj} + \sum_{i:(i,j)\in E} f_{ij} - \sum_{i:(i,j)\in E} f_{ji} = 1 \qquad \forall j \in V, \tag{4.12b}$$

$$f_{ij} \leq M \bar{z}_{l(i,j)} \quad \forall (i,j) \in E, \tag{4.12c}$$

$$f_{ji} \leq M \bar{z}_{l(i,j)} \quad \forall (i,j) \in E, \tag{4.12d}$$

$$f_{sj} \leq M \bar{z}_{l(s,j)} \quad \forall j \in V, \tag{4.12e}$$

$$f_{ij}, f_{ji} \geq 0 \qquad \forall ij \in E, \tag{4.12f}$$

$$f_{sj} \geq 0 \qquad \forall j \in V. \tag{4.12g}$$

Associando variáveis duais μ às restrições (4.12b) e $\lambda \leq 0$ às restrições (4.12c)-

(4.12e), expressa-se o dual de $P(\bar{z})$ como:

$$D(\bar{z}) \max \quad \sum_{(i,j)\in E} M\bar{z}_{l(i,j)}(\lambda_{ij}+\lambda_{ji}) + \sum_{j\in V} M\bar{z}_{l(s,j)}\lambda_{sj} + \sum_{j\in V}\mu_j \tag{4.13a}$$

$$\text{s.a.} \quad \lambda_{ij}+\mu_j-\mu_i \leq 0 \quad \forall (i,j)\in E, \tag{4.13b}$$

$$\lambda_{ji}-\mu_j+\mu_i \leq 0 \quad \forall (i,j)\in E, \tag{4.13c}$$

$$\lambda_{sj}+\mu_j \leq 0 \quad \forall j \in V, \tag{4.13d}$$

$$\lambda_{ij},\lambda_{ji} \leq 0 \quad \forall (i,j)\in E, \tag{4.13e}$$

$$\lambda_{sj} \leq 0, \mu_j \text{ livre} \quad \forall j \in V \tag{4.13f}$$

Por fim, apresenta-se o problema mestre relativo à decomposição de Benders em (P). Sejam $\mathscr{J}$ e $\mathscr{K}$ os conjuntos de vértices e direções extremas do conjunto viável do problema dual, respectivamente.

$$\begin{aligned}
&(Mestre)\min \quad \sum_{l\in C} z_l + x \\
&\text{s.a} \\
&x \geq \sum_{(i,j)\in E} Mz_{l(i,j)}(\bar{\lambda}_{ij}+\bar{\lambda}_{ji}) + \sum_{j\in V} Mz_{l(s,j)}\bar{\lambda}_{sj} + \sum_{j\in V}\bar{\mu}_j, \quad \forall (\bar{\lambda},\bar{\mu}) \in \mathscr{J} \\
&\sum_{(i,j)\in E} Mz_{l(i,j)}(\bar{\lambda}_{ij}+\bar{\lambda}_{ji}) + \sum_{j\in V} Mz_{l(s,j)}\bar{\lambda}_{sj} + \sum_{j\in V}\bar{\mu}_j \leq 0, \quad \forall (\bar{\lambda},\bar{\mu}) \in \mathscr{K} \\
&\sum_{l\in L} z_l \leq k \\
&z_l \in \{0,1\} \quad \forall l \in L\cup C.
\end{aligned} \tag{4.14}$$

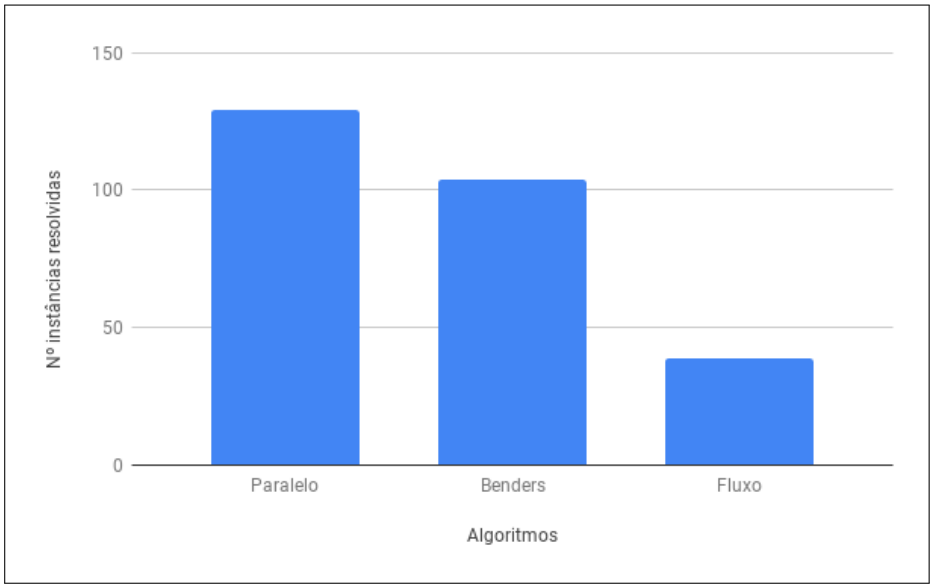
Por padrão, as versões do *Cplex* 12.8 e posteriores possuem uma decomposição de Benders automatizada. Ela foi aplicada com as mesmas estratégias usadas para a resolução do modelo sem a decomposição. Porém, o problema mestre inicial gerava uma relaxação muito fraca. Por isso, as desigualdades (4.7) são usadas para fortalecer o problema.

Os resultados detalhados dos experimentos encontram-se na Tabela 6 do Apêndice A. Devido ao uso da decomposição de Benders automatizada, o limite de tempo foi aumentado para 7500s para compensar o *presolve* usado pelo *Cplex* nesse modo de resolução. Algumas instâncias iniciais menores possuem tempo limite em 7300s. Apesar de usar configurações de resolução do teste similares às usadas para modelo de fluxo anterior, o *hardware* agora possui 16GB em vez de 8GB. Além dessa mudança, também foi modificado o modo paralelo de resolução para o modo oportunístico.

No modo oportunístico, o algoritmo de busca relaxa a sincronização entre os nós, garantindo maior liberdade na ordem de exploração dos mesmos. Como o objetivo desta seção é mostrar os melhores resultados, esse modo é ativado. Por outro lado, ao usar esse modo, os tempos de resolução possuem uma maior variância.

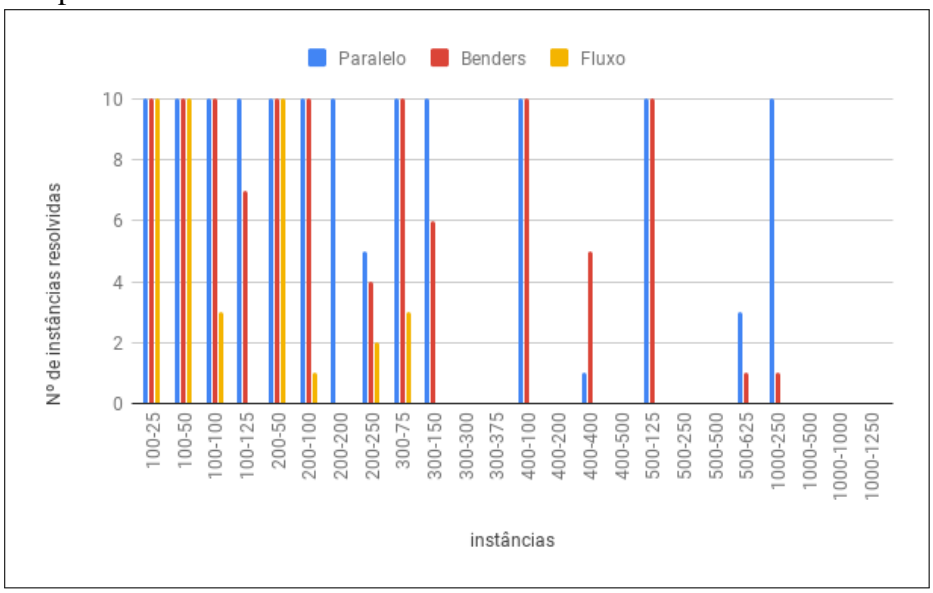
Como feito nos testes anteriores, são comparados os resultados do algoritmo enumerativo paralelo, o *Branch&Cut* no modelo de FG*k*R_Fluxo e com o modelo de fluxo usando Benders. Como esperado, usando a decomposição, um maior número de instâncias são resolvidas dentro do tempo limite, vide Figura 20. Apesar de não resolver mais instâncias que o enumerativo, é possível ver, com mais detalhe na Figura 21, que algumas instâncias com 400 vértices e 400 cores são resolvidas a mais que o paralelo. Também deve ser chamado a atenção, na Tabela 6, que certas instâncias alcançam o valor objetivo ótimo, porém não se consegue provar a otimalidade no tempo limite, fato pertinente não somente a esse modelo. Por fim, na Figura 22 são mostrados os tempos médios de resolução, onde se observam grandes reduções de tempo depois da utilização da decomposição.

Figura 20 – Número de instâncias resolvidas comparativo com Modelo de Fluxo por Benders



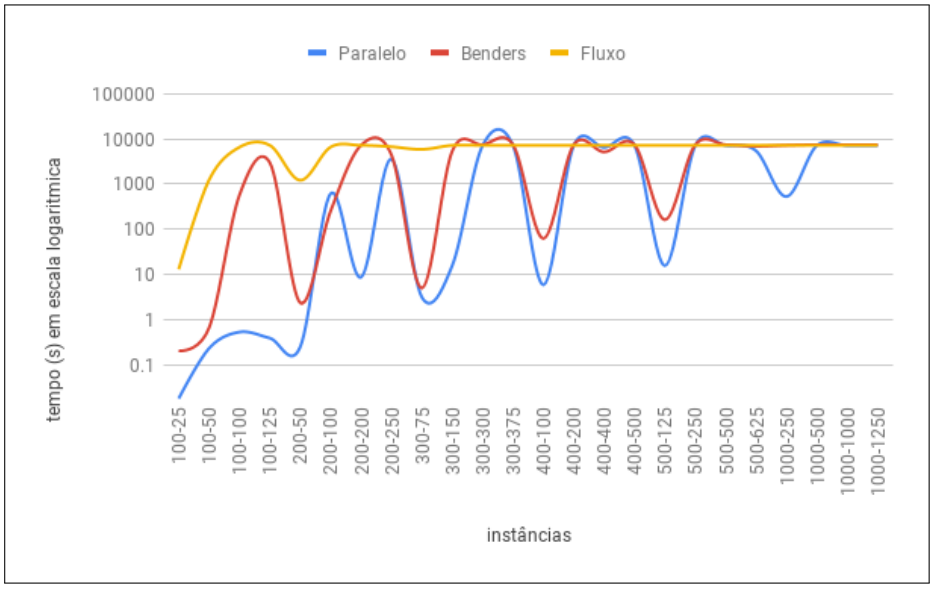
Fonte: elaborado pelo autor (2020).

Figura 21 – Número de instâncias resolvidas comparativo com Modelo de Fluxo por Benders separado



Fonte: elaborado pelo autor (2020).

Figura 22 – Tempo médio de resolução das instâncias comparativo com Modelo de Fluxo por Benders separado



Fonte: elaborado pelo autor (2020).

5 MELHORES ALGORITMOS

Neste capítulo são apresentadas as modificações finais aplicadas aos métodos de resolução em testes, avaliadas com instâncias geradas a partir de variações nos parâmetros usados na literatura para o FG*k*R (CERULLI *et al.*, 2014), porém presentes em outros testes para problemas parecidos (SILVA, 2018). Além disso, a avaliação de desempenho é feita também através de outras métricas, como número de nós explorados e profundidade da árvore de busca.

5.1 Estratégias

Nesta seção serão exploradas estratégias adicionais que foram utilizadas para os novos testes.

5.1.1 Modo oportunístico

Como visto no capítulo anterior, uma das estratégias para melhorar os tempos de computação é ativar o modo oportunístico. Diferentemente do modo determinístico padrão, no modo oportunístico a velocidade de exploração de nós é maior. Por outro lado, possui algumas inconsistências, como: maior fronteira de nós candidatos a serem explorados, maior variância no tempo de execução e desbalanceamento da árvore de busca. Como resultado, dois casos podem acontecer. No primeiro, soluções são alcançadas de forma mais rápida, ou no caso do segundo, são feitas "péssimas" escolhas durante o *branching* e o tempo de execução alcança seu limite, ou um valor além do esperado. O que se espera é que o modo oportunístico com as heurísticas do *Cplex*, o ganho advindo do paralelismo e as relaxações lineares geradas sejam boas o suficiente para minimizar esses distúrbios de tempo.

5.1.2 Nova regra de Branching

A ideia principal sobre *Branching* em variáveis é fixar em 0 ou 1 uma variável com valor fracionário na relaxação linear, particionando o problema em dois. No caso das variáveis binárias relativas a cor, como discutido em (SILVA, 2018), o "peso" de fixar uma variável binária de cor em 0 é muito menor que fixar em 1. O fato é que, quando se possui um conjunto de cores vasto, fazer com que uma variável de cor seja zero não causa grande impacto no problema. Por outro lado, fixar uma cor em 1, ou seja, obrigar que ela faça parte das soluções viáveis enraizadas

no nó atual, significa que, para aquela subárvore, é necessário encontrar apenas mais $k-1$ cores (FGkR em $(G,k-1)$) e, possivelmente, que existe um subconjunto de cores dominadas que não precisam ser exploradas em relação à nova solução parcial encontrada.

Os efeitos de uma regra de *branching* se somam ou são interdependentes daqueles de outros elementos aplicados durante o processo de *Brach&Cut*. Nessa perspectiva, chama-se atenção para ideias propostas no capítulo anterior que precisam ser avaliados em conjunto com a escolha da regra de *branching*. Dentre elas, destacam-se a poda usando limite superior provido pela heurística gulosa, a poda usando o limite inferior provido em Proposição 2.5.1 que é aplicada no algoritmo enumerativo e por fim a utilização de prioridades para variáveis de cor com maior número de arestas. Com relação às podas, veja que, no momento que uma solução parcial é eliminada, o processo de ramificação que ocorreria naquele nó não será mais necessário.

Durante testes computacionais foi visto que a escolha da variável a ser ramificada tem um impacto grande (positivo ou negativo) no desempenho dos algoritmos, o que leva à questão sobre a existência de uma ordem de prioridade, pré-estabelecida, mais eficiente que a usada preliminarmente. Como esperado de um problema NP-Difícil, questões como essa são de difícil resposta. Na verdade, parece mais plausível que a decisão sobre a variável a ser ramificada seja melhor tomada a partir de informações coletadas ao longo do processo de busca.

Nessa perspectiva, com intuito de melhorar o processo de ramificação, em vez de escolher uma ordem de prioridades inicial, tenta-se utilizar informações da solução atual para gerar uma ordem mais dinâmica. Por exemplo, veja que, quando uma solução parcial está a um nível de profundidade de uma solução com k cores, ou seja, falta somente uma cor a ser incluída nessa para completar o máximo possível de cores, esse subproblema resume-se a fazer uma busca linear pela cor não usada (não dominada) que, ao ser adicionada à solução parcial, induz o menor número de componentes. Estendendo esse raciocínio, o seguinte critério de ramificação é definido:

1. Construir a solução parcial viável $\bar{z} \in \{0,1\}^{|L|}$ (variáveis de cor do grafo original), baseada no nó atual, fixando em 1 as variáveis que são limitadas inferiormente por 1 ($z_i \geq 1$) no processo de ramificação. Logo, as demais variáveis são fixadas em 0.
2. Encontrar $I \subseteq L$ o conjunto de variáveis de cor do grafo original G as quais possuem valores de relaxação linear fracionários no nó atual e $\bar{L} = \{c \in L : \bar{z}_c = 1\}$.
3. Aplicar ramificação ($z_i \geq 1$ ou $z_i \leq 0$) na cor $i \in I$ tal que $\mathscr{W}(G[\bar{L} \cup \{i\}]) \leq \mathscr{W}(G[L \cup j]), \forall j \in I$, ou equivalentemente, $i \in \arg\max_{j \in I}(\mathscr{W}(G[\bar{L}]) - \mathscr{W}(G[\bar{L} \cup \{j\}])$.

4. Caso não possa fixar mais cores em 1 (por causa de k) ou $I = \emptyset$, deixar o *Cplex* escolher a variável de cor adicional em C a ser ramificada.

observar a estreita ralação entre os cálculos necessários para aplicação do item 3 acima e de outros ingredientes já discutidos anteriormente. Por exemplo, o procedimento para fazer poda por limite inferior, derivado da Proposição 2.5.1, já calcula a diferença $(\mathscr{W}(G[\bar{L}]) - \mathscr{W}(G[\bar{L} \cup \{j\}])$, que determina a redução no número de componentes de uma solução parcial $\bar{L} \subseteq L$ com a inclusão de uma cor $j \in L \setminus \bar{L}$. Além disso, quando $\mathscr{W}(G[\bar{L}]) - \mathscr{W}(G[\bar{L} \cup \{j\}]) = 0$, a cor j é dominada por $\bar{L}$ (Definição 3.1.7) e pode ser descartada. Dessa forma, a partir desses cálculos comuns, adiciona-se um *BranchCallback*, usando o método de *prune*, para aplicar poda por limite inferior, e um *makeBranch* para fixar cores dominadas pela solução parcial atual em zero ao mesmo tempo em que aplica ramificação na variável que induz o menor número de componentes.

5.1.3 Fortalecendo Relaxação linear

No processo de fortalecimento dos modelos usados, em vez de escolher aplicar somente as desigualdades (4.7) ou alternativamente (4.8) nos modelos iniciais, ambos os tipos de desigualdades válidas são aplicados, principalmente para fortalecer o modelo mestre de fluxo usando decomposição.

Um subconjunto de desigualdades do tipo (4.4) foi testado. Essas são geradas por um procedimento similar ao kMVCA, no intuito de escolher soluções parciais, e acrescentadas ao modelo inicial. Porém, esse fortalecimento do modelo inicial não levou a um melhor desempenho nos testes efetuados (modo determinístico e oportunístico).

5.1.4 Perturbação limitada

O processo de pertubação na geração de desigualdades de cortes coloridos (subseção 4.2.2) se mostrou promissor em testes do capítulo anterior. Porém não é trivial, previamente à resolução das instâncias, saber quais dessas desigualdades são melhores. Não somente isso, mas existe o custo de tempo para gerar essas em ambos *LazyContraintCallback* e *UserCutCallback*. Para tanto, o que é proposto aqui é limitar o escopo do processo de pertubação ao *UserCutCallback*, em conjunto com um critério de parada adicional, limitando a k o número de cores do grafo original ativadas no grafo reduzido. Veja que as desigualdades de cortes coloridos usadas no *LazyContraintCallback* sem pertubação são suficientes para separação da solução inteira inviável.

5.2 Experimentos Computacionais

Nesta seção são vistos os parâmetros utilizados para geração de instâncias. Em seguida, são explorados os métodos de resolução usados e dos resultados encontrados.

5.2.1 Geração de instâncias

Os testes são novamente baseados em (CERULLI *et al.*, 2014) e na Subseção 4.3.2, porém com algumas mudanças. Cada grafo é gerado com os seguintes parâmetros adicionais: $m(G) = d(G)\frac{n(G)\cdot(n(G)-1)}{2}$, $d(G) = \{0.2, 0.4, 0.6\}$, obtendo um banco de testes (600 instâncias) com grafos com densidades maiores que 0.2. O valor de k é determinado como nas instâncias do capítulo anterior, conforme (CERULLI *et al.*, 2014).

Desse banco, 43 instâncias são escolhidas, aquelas onde o valor de k é pelo menos 2 e, portanto, não possuem solução trivial. Vale destacar que esse grupo selecionado mantém a diversidade dos parâmetros usados na geração.

5.2.2 Configurações dos métodos

Nesta subseção são descritos os modelos usados, as configurações do *Branch&Cut* e do *Cplex*, definindo as seguintes abordagens de solução, identificadas (em parênteses) conforme apresentação nas tabelas de resultados:

- Algoritmo de *backtracking* paralelo (Enumerativo Paralelo) : Este usa o mesmo processo algorítmico da subseção 4.4.1 com alguns ajuste de programação e com um melhor limite superior, fornecido pelo *k*MVCA.
- Modelo de fluxo usando Benders (Benders): Baseado na formulação da subseção 4.1.3 para o grafo estendido, adicionando as desigualdades (4.7) e (4.8) ao problema mestre. Usa a Decomposição de Benders automatizada do *Cplex* e prioridade de ramificação para variáveis de cores com maior número de arestas (essa estratégia sem essa priorização piorou o desempenho médio nos testes).
- Abordagem baseada no *B&C-Ccut2* (*Ccut2-Modificado*): Similar ao *B&C-Ccut2*, porém usando apenas as desigualdades (4.7) para fortalecer o modelo inicial.
- Abordagens baseadas no *B&C-Ccut2*, usando ambas (4.7) e (4.8) no modelo inicial.
  - (*Ccut3*) : usa poda por limite inferior, fixação de variáveis por dominância e heurística gulosa para fazer *branching* de variáveis fracionárias num *BranchCallback*

(subseção 5.1.2).

– (*Ccut4*) : similar ao Ccut3, porém com limitação do processo de pertubação (subseção 5.1.4).

Os modelos são executados no *Cplex* 12.8 em modo oportunístico com 8 *threads*, com limite superior (*UpperCutoff*) provido pelo *k*MVCA e maior ênfase para busca por soluções inteiras. O método de *presolve* é desligado em todos os modelos, e métodos de pré-processamento para quebra de ciclos não foram utilizados. O tempo limite para resolução de cada instância foi estipulado em 20 minutos (1200s) para cada método, até mesmo o modelo de fluxo, dado que serão comparados tempos médios.

5.2.3 Resultados

Os resultados desta subseção são expostos em termos de médias de parâmetros de tempo, em segundos, e número de instâncias resolvidas. Além disso, é possível analisar a distância relativa entre os valores da relaxação linear de cada modelo e das soluções viáveis por ele obtidas através do *GAP* percentual, calculado da seguinte forma:

$$GAP = 100\frac{(OBJ - LI)}{OBJ}, \tag{5.1}$$

onde *LI* é o melhor limite inferior dado pela formulação, e *OBJ* é o melhor valor objetivo de solução inteira encontrada.

Antes de apresentar os resultados dos melhores métodos em geral, um comparativo de algumas das abordagens utilizando a formulação (FG*k*R_CCut) é feito. Na Tabela 1 são apresentados os tempos (t(s)), melhor valor de função objetivo de solução viável (OBJ) , o *GAP* e o número de nós explorados da árvore de *Branch&Cut* (#nós). Em termos de tempo médio o *Ccut4* leva 377s por instância, sendo mais rápido que os *Ccut2-Modificado* (489s) e *Ccut3* (422s).

É interessante ver que, com o novo processo de *branching*, explora-se um conjunto de nós mais restrito. Em média no *Ccut2-Modificado* (preferência de *branching* em cores com mais arestas) são explorados mais nós (280995, em média) durante o tempo de execução que nos outros dois (<82000). Ao comparar instâncias, pode-se considerar que em geral as relaxações de *Ccut3* e *Ccut4* são mais fortes, o que era esperado (subseção 5.1.3). Inesperado, porém, é que em instâncias como 500-24950-500-7-3 ou 300-8970-375-9-7 o *GAP* para *Ccut4* seja muito menor que para *Ccut3*, dado que a única diferença é a limitação do número de iterações no processo de

pertubação. Ressalve-se que, para esses casos, os limites inferiores são bem maiores que 1, o que não é tão comum para o problema, de acordo com todos os testes feitos.

Essas características de desempenho mostram por que o *Ccut4* consegue encontrar mais soluções ótimas (31 de 43) neste conjunto de testes (*Ccut3* resolve 30 de 43).

Tabela 1 – Comparativos das abordagens baseadas em FG*k*R_Ccut

	CCut2-Modificado				CCut3				CCut4			
instancias	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*
100-2475-100-3-7	0.21	9	1257	0%	0.3	9	373	0%	0.08	9	145	0%
100-3960-125-3-1	0.6	3	3294	0%	0.97	3	2808	0%	1.14	3	2460	0%
100-990-100-6-5	9.45	13	35247	0%	0.55	13	2103	0%	0.2	13	967	0%
100-990-125-6-5	2.63	17	12287	0%	0.14	17	233	0%	0.05	17	129	0%
100-990-50-6-6	0.02	1	7	0%	0.01	1	0	0%	0	1	0	0%
200-15920-200-3-5	13.83	3	23310	0%	27.43	3	19642	0%	27.59	3	20176	0%
200-15920-250-3-6	20.66	11	30455	0%	34.74	11	21897	0%	33.05	11	19545	0%
200-3980-100-6-5	91.06	2	234271	0%	102.38	2	196585	0%	100.33	2	193104	0%
200-3980-200-6-7	1202.38	37	509834	90.41%	3.88	36	3339	0%	0.67	36	1101	0%
200-3980-250-12-4	1.17	1	2838	0%	1.35	1	1526	0%	0.29	1	91	0%
200-3980-50-3-6	0.24	6	970	0%	0.38	6	933	0%	0.35	6	931	0%
200-9950-100-3-8	0.08	1	79	0%	0.14	1	66	0%	0.2	1	150	0%
200-9950-200-3-3	6.2	26	15264	0%	4.24	26	2083	0%	1.02	26	645	0%
200-9950-250-6-7	0.23	1	284	0%	1.88	1	1204	0%	1.05	1	617	0%
300-22425-150-2-6	1.05	23	286	0%	3.87	23	243	0%	2.71	23	241	0%
300-22425-300-4-3	734.25	14	635842	0%	735.39	14	386444	0%	645.3	14	352227	0%
300-22425-375-4-3	815.07	29	646216	0%	384.19	29	143891	0%	233.91	29	99119	0%
300-22425-75-2-8	0.05	1	11	0%	0.09	1	9	0%	0.16	1	23	0%
300-35880-150-2-4	0.87	3	288	0%	3	3	285	0%	3.18	3	287	0%
300-35880-300-2-7	7.67	41	620	0%	12.56	41	269	0%	1.62	41	89	0%
300-35880-375-4-3	1200.01	2	683660	50%	1200.05	2	225744	50%	1200.04	3	247061	66.67%
300-8970-150-4-3	56.53	37	119816	0%	18.55	37	17707	0%	10.51	37	10445	0%
300-8970-300-9-1	1203.23	16	1440740	97.75%	1200.01	11	454575	90.91%	1200.02	11	588119	90.91%
300-8970-375-9-7	1202.94	30	984556	96.66%	1200.01	30	132179	96.67%	1200.02	29	298609	55.68%
300-8970-75-4-7	4.41	1	11434	0%	0.1	1	20	0%	0.22	1	188	0%
400-15960-100-3-3	6.97	15	7820	0%	10.13	15	7110	0%	10.78	15	7110	0%
400-15960-200-6-4	1202.8	8	1319751	87.50%	1200.01	7	561721	85.71%	1200.01	8	545421	87.50%
400-15960-400-12-6	1207.87	3	895979	66.66%	4.21	1	613	0%	15.61	1	2918	0%

Continua próxima pagina

Tabela 1 – continuação pagina anterior

	CCut2-modified				CCut3				CCut4			
instancias	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*
400-15960-500-12-2	1201.84	13	988640	92.30%	1200.05	6	231366	83.33%	1200.01	9	287753	88.89%
400-39900-200-3-5	51.07	5	33358	0%	95.4	5	32745	0%	96.03	5	32695	0%
400-39900-400-3-3	214.44	60	110489	0%	94.3	60	9847	0%	13.47	60	2010	0%
400-39900-500-6-4	1202.08	6	735924	83.33%	1200.08	4	146843	75.00%	1200.07	5	155506	80.00%
400-63840-400-3-6	567.38	12	138211	0%	1200.1	12	116141	91.67%	1200.07	12	64446	91.67%
400-63840-500-3-7	864.4	31	203243	0%	1200.1	31	51825	96.77%	1200.12	31	62055	96.23%
500-24950-125-3-7	20.15	21	12203	0%	26.84	21	11826	0%	30.54	21	11817	0%
500-24950-250-7-3	1202	2	656419	50%	1200.04	2	354030	50%	1200.03	2	255327	50%
500-24950-500-7-3	1201.6	81	346111	98.76%	1200.09	77	34040	98.70%	1200.06	72	115432	28.74%
500-24950-625-7-3	1201.57	118	263498	99.50%	304.92	114	18475	0%	18.69	114	3641	0%
500-62375-250-3-5	154.29	6	55678	0%	369.11	6	53603	0%	403.06	6	53284	0%
500-62375-500-3-5	564.82	81	188543	0%	330.92	81	14583	0%	40.58	81	3463	0%
500-62375-625-7-6	1201.75	4	424882	75.00%	1200.18	2	55848	50.00%	131.62	1	5587	0%
500-99800-500-3-6	1200.01	15	158182	93.33%	1200.12	15	38698	93.33%	1200.36	17	40003	94.12%
500-99800-625-3-1	1200.03	45	150961	97.77%	1200.35	43	26240	97.67%	1200.31	41	24636	97.56%
médias	489.30	19.86	280995	27%	422.63	19.16	78598	25%	377.33	19.14	81618	22%

Dados os melhores resultados do *Ccut4*, esse é comparado com os métodos de *backtracking* paralelo e o baseado no modelo de fluxo usando Benders (Tabela 2), como definidos na subseção anterior. Diferentemente dos modelos, o método paralelo não produz um *GAP* ou número de nós explorados. Por outro lado, usa-se de um campo referente à profundidade (d) onde o algoritmo enumerativo encontra a melhor solução, este é zero quando a melhor solução é a encontrada pelo *k*MVCA.

É interessante ver que o algoritmo paralelo consegue descer a níveis bem baixos (220 de 300) dentro do tempo limite, uma vantagem provavelmente atribuída à sua velocidade de exploração, dada a pouca granularidade no processamento de seus nós. O que foi visto, tanto em testes anteriores quanto nesses, é que, quando os modelos, usando a mesma prioridade de variáveis de cores, não conseguem gerar limites inferiores bons (pelo menos maiores que 1) o paralelismo do *backtracking* é mais propício a gerar bons resultados.

Na Tabela 2, os valores objetivos (OBJ) do método enumerativo são ótimos, a não ser que venha acompanhado de um caractere '*'. À exceção de um, todos os limites superiores (OBJ) encontrados pelo *Ccut4* são melhores ou iguais aos encontradas pelos outros dois métodos. A única exceção ocorre na instância 500-99800-500-3-6, onde o algoritmo paralelo é melhor que ambos *Benders* e *Ccut4*, nos quais os limites inferiores é a solução trivial conexa. Vale destacar que, das 43 instâncias, o algoritmo de *backtracking*, o modelo de fluxo usando Benders e o Ccut4 encontram soluções ótimas para 29, 25 e 31 delas, respectivamente.

Em termos gerais, a decomposição de Benders não conseguiu ser o suficiente para fazer o modelo de fluxo ter um desempenho melhor que o por cortes coloridos. Talvez por motivos de limitações impostas pela decomposição automatizada pelo *Cplex* ou por, principalmente, ao longo deste trabalho, não terem sido encontradas desigualdades válidas relacionando variáveis de cores e de fluxo "boas" o suficiente para gerar planos de cortes profundos. Mesmo assim, vale ressaltar, em comparação ao enumerativo, que o tempo médio para o Benders (552s) é bem próximo, com uma diferença de aproximadamente 51s. Posto que Benders resolve otimamente 4 instâncias a menos, nas quais atinge o tempo limite, essa pequena diferença nos tempos médios mostra que a decomposição consegue um desempenho bem próximo nas instâncias em que ambos encontram o ótimo.

Como visto nas tabelas 1 e 2, o algoritmo *Cut4* obteve melhor desempenho em geral. O que deve ser chamado a atenção é como o novo processo de *branching* em conjunto com a geração dos planos de cortes usando pertubação limitada oferecem *GAP's* melhores.

Tabela 2 – Comparativos melhores algoritmos

	Enumerativo Paralelo			Benders				CCut4			
instancias	d	t (s)	OBJ	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*
100-2475-100-3-7	10	0.0301741	9	0.81	9	646	0%	0.08	9	145	0%
100-3960-125-3-1	31	0.931365	3	2.79	3	8053	0%	1.14	3	2460	0%
100-990-100-6-5	16	0.0780713	13	1.23	13	2401	0%	0.2	13	967	0%
100-990-125-6-5	0	0.0100668	17	0.35	17	767	0%	0.05	17	129	0%
100-990-50-6-6	42	0.127937	1	0.07	1	0	0%	0	1	0	0%
200-15920-200-3-5	0	35.2999	3	25.91	3	40646	0%	27.59	3	20176	0%
200-15920-250-3-6	57	43.1746	11	366.71	11	191663	0%	33.05	11	19545	0%
200-3980-100-6-5	73	1072.01	2	140.93	2	359690	0%	100.33	2	193104	0%
200-3980-200-6-7	15	0.643152	36	8.91	36	4666	0%	0.67	36	1101	0%

Continua próxima pagina

Tabela 2 – continuação pagina anterior

		Enumerativo Paralelo		Benders				CCut4			
instancias	d	t (s)	OBJ	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*
200-3980-250-12-4	0	1200	2*	13.64	1	35755	0%	0.29	1	91	0%
200-3980-50-3-6	13	0.18985	6	2.32	6	5716	0%	0.35	6	931	0%
200-9950-100-3-8	86	0.385941	1	1.57	1	855	0%	0.2	1	150	0%
200-9950-200-3-3	0	1.00793	26	12.16	26	2115	0%	1.02	26	645	0%
200-9950-250-6-7	142	68.0378	1	1.08	1	166	0%	1.05	1	617	0%
300-22425-150-2-6	0	1.28159	23	17.96	23	6960	0%	2.71	23	241	0%
300-22425-300-4-3	220	1200	14*	1200.02	15	133948	93.33%	645.3	14	352227	0%
300-22425-375-4-3	49	693.333	29	1200.04	29	49267	96.55%	233.91	29	99119	0%
300-22425-75-2-8	8	0.0826843	1	1.94	1	0	0%	0.16	1	23	0%
300-35880-150-2-4	0	2.01138	3	14.44	3	2926	0%	3.18	3	287	0%
300-35880-300-2-7	0	2.0789	41	69.79	41	1982	0%	1.62	41	89	0%
300-35880-375-4-3	51	1200	3*	1200	4	994502	75%	1200.04	3	247061	66.67%
300-8970-150-4-3	0	13.609	37	226.65	37	29539	0%	10.51	37	10445	0%
300-8970-300-9-1	0	1200	16*	1200.01	16	1147588	93.75%	1200.02	11	588119	90.91%
300-8970-375-9-7	0	1200	30*	1200.02	30	186237	86.38%	1200.02	29	298609	55.68%
300-8970-75-4-7	21	0.358696	1	3.08	1	3734	0%	0.22	1	188	0%
400-15960-100-3-3	0	5.38892	15	15.77	15	13867	0%	10.78	15	7110	0%
400-15960-200-6-4	0	1200	8*	1200.01	8	1370579	87.50%	1200.01	8	545421	87.50%
400-15960-400-12-6	0	1200	3*	1203.24	3	1415186	66.66%	15.61	1	2918	0%
400-15960-500-12-2	0	1200	13	1200.02	13	2212737	92.30%	1200.01	9	287753	88.89%
400-39900-200-3-5	26	89.0705	5	124.67	5	66097	0%	96.03	5	32695	0%
400-39900-400-3-3	16	24.7576	60	790.71	60	13184	0%	13.47	60	2010	0%
400-39900-500-6-4	0	1200	6*	1200	6	904349	83.33%	1200.07	5	155506	80.00%
400-63840-400-3-6	117	1102.57	12	1200	13	287596	92.30%	1200.07	12	64446	91.67%
400-63840-500-3-7	45	1200	31*	1200	32	51716	96.87%	1200.12	31	62055	96.23%
500-24950-125-3-7	11	15.4353	21	74.49	21	30960	0%	30.54	21	11817	0%
500-24950-250-7-3	0	1200	2*	1203.57	2	1255660	50%	1200.03	2	255327	50%
500-24950-500-7-3	0	1200	81*	1200.08	76	58645	48.07%	1200.06	72	115432	28.74%
500-24950-625-7-3	9	57.7161	114	1200.09	114	38386	10.80%	18.69	114	3641	0%
500-62375-250-3-5	17	268.406	6	238.42	6	77167	0%	403.06	6	53284	0%
500-62375-500-3-5	0	79.0268	81	1200.05	81	10999	44.66%	40.58	81	3463	0%
500-62375-625-7-6	0	1200	5*	1200	3	407339	66.67%	131.62	1	5587	0%
500-99800-500-3-6	65	1200	15*	1199.98	20	170022	95.00%	1200.36	17	40003	94.12%
500-99800-625-3-1	32	1200	41*	1200.01	41	32763	97.56%	1200.31	41	24636	97.56%

Continua próxima pagina

Tabela 2 – continuação pagina anterior

	Enumerativo Paralelo			Benders				CCut4			
instancias	d	t (s)	OBJ	t (s)	OBJ	#nós	*GAP*	t (s)	OBJ	#nós	*GAP*
médias	28	501.79	19.72	552.64	19.74	270398	32%	377.33	19.14	81618	22%

6 CONCLUSÕES E TRABALHOS FUTUROS

Problemas de otimização combinatória em grafos coloridos arestas têm propiciado bastante pesquisa na última década. Em particular, (SILVA, 2018) mostra um conjunto de problemas com ênfase em conectividade nesse tipo de grafos. Dentre eles, destacou-se o problema da FGkR, dado seu grau de dificuldade (NP-Difícil), por possuir ainda pouca literatura a ele dedicada e, sobretudo, por ter sido proposto até aqui um único método exato (de certa forma, uma busca exaustiva simples via *backtracking*).

O intuito geral deste trabalho foi propor uma formulação matemática e um método de resolução que se mostrasse mais promissor em relação ao método de *backtracking*. Nesse sentido, foi obtido sucesso. De fato, foram introduzidas três formulações para o problema, dentre outras também propostas, mas que não estejam presentes neste trabalho. Essas formulações fundamentam-se em caracterizações do problema aqui apresentadas. Elas foram a base para algumas abordagens de solução exata, que superaram o algoritmo enumerativo da literatura em vários testes realizados.

É correto afirmar que aplicar o método de *Branch&Cut* padrão do *Cplex* seria o "suficiente" para resolver o problema à otimalidade, no sentido de que eventualmente ele chegaria a uma solução ótima. No entanto, devido à complexidade do problema e às características dos modelos propostos, dificilmente tal processo padrão é capaz de produzir uma solução exata em tempo de processamento aceitável.

Para tanto, foram estudadas e avaliadas diferentes estratégias de planos de corte, separação de desigualdades de cortes coloridos, ramificação, pré-processamento para redução de dificuldade das instâncias, geração de desigualdades válidas e modificações de configurações próprias do *Cplex*. As combinações que mostraram melhores desempenhos em testes foram expostas no trabalho.

Como uma das implicações de o problema da FGkR ser *NP-Difícil*, apontar qual método é o melhor para resolvê-lo (ou mesmo para certo tipo de instância) não é tarefa fácil, ainda que sejam realizados exaustivos testes computacionais. Essa conclusão tornou-se inclusive mais obscura, tendo em vista que nos testes observaram-se instâncias que são resolvidas em tempos maiores do que o esperado, em comparação com outras instâncias, dadas as magnitudes dos parâmetros ($|E|$, $|L|$ e k). O que se pode recomendar é a utilização do modelo FGkR_CCut (*Ccut4* em geral) e, quando não for possível encontrar "boas" relaxações lineares, o método de *backtracking* paralelo, com mais ênfase no segundo em ambientes computacionais mais robustos

em termos de paralelismo e uso de ferramentas de programação paralela menos genéricas.

No que diz respeito a aspectos teóricos, no Capítulo 3 foram formalizadas ideias em relação ao problema e suas formulações. Como resultado são estabelecidas caracterizações do problema em relação a ideias já existentes na literatura para outros problemas (SILVA, 2018), por exemplo, cortes coloridos e dominância. Também são estudadas novas características mais especializadas, que não somente motivam as formulações, como enriquecem a literatura do problema. Além disso, foram identificadas classes de GCA's onde é possível resolver a FGkR (G,k) de forma exata em tempo polinomial. Tratam-se de grafos os quais possuem estruturas bem definidas, que são fáceis de verificar numa instância arbitrária.

Espera-se que, com as formulações providas no Capítulo 4, seja possível motivar o estudo de métodos exatos para o problema. Algumas abordagens usando essas formulações foram avaliadas neste trabalho, mas há diversas outras possibilidades que podem ser tentadas, especialmente no que diz respeito a estratégias de introdução das desigualdades que definem os modelos ou de desigualdades válidas que possam fortalecê-los. Em particular, apesar de não ter apresentado desempenhos bons da forma como foi implementada, a formulação FGkR_Floresta parece ser promissora, considerando que ela é definida diretamente sobre o grafo original. Estudos iniciais usando desigualdades MTZ (MILLER *et al.*, 1960) para quebra de ciclos foram feitos, porém sem aplicação de desigualdades válidas customizadas (provavelmente, mais fortes) para o problema. Comentário similar também pode ser feito em relação à formulação por fluxo. A decomposição de Benders conseguiu melhorar o desempenho mas não foi o suficiente. Talvez desigualdades válidas que relacionem fluxo com cores sejam necessárias para contrabalançar o Big-M da formulação, ou pelo menos limitar a constante M.

Durante os testes, observou-se que encontrar bons limites inferiores é uma das principais dificuldades para a resolução do problema. Mesmo desigualdades como (4.7) e (4.8) não foram suficientes para garantir bons limites inferiores. O mesmo ocorre para as desigualdades válidas da subseção 4.1.4, cujo efeito foi possivelmente limitado pela maneira como foram usadas. Acredita-se que encontrar desigualdades válidas para fortalecer os modelos ou alguma forma de encontrar melhores limites inferiores (por exemplo, relaxação lagrangiana), gerando limites inferiores ao menos diferentes daquele dado pelo grafo conexo trivial (grafos de entrada são todos conexos), seja importante para gerar métodos de solução exatos mais eficientes.

Ainda sobre o Capítulo 4, com respeito a técnicas de pré-processamento (procedimento de quebra de ciclos e transformações) é observado que estas não necessariamente

melhoram o desempenho dos modelos. Por outro lado, vale ressalvar que existem ganhos para o método de *backtracking*.

Ao longo deste trabalho, focou-se em instâncias geradas de forma aleatória. É interessante notar que em cenários reais existem padrões inerentes à aplicação (como estruturas de grafos em grade). Estes possuem caracterizações específicas fortes que podem permitir o desenvolvimento de métodos exatos menos genéricos. Talvez em grafos bem estruturados seja possível estabelecer noções melhores sobre ordem de prioridade para variáveis de cores e o impacto de transformações sobre o grafo de entrada.

REFERÊNCIAS

BAZARAA, M. S.; JARRIS, J. J.; SHERALI, H. D. **Linear programming and network flows**. 4. ed. Hoboken, New Jersey: J. Wiley, 2010. ISBN 978-0-470-46272-0.

BENDERS, J. F. Partitioning procedures for solving mixed-variables programming problems. **Numer. Math.**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 4, n. 1, p. 238–252, dez. 1962. ISSN 0029-599X.

BONDY, J. A. **Graph theory**. New York: Springer, 2010. ISBN 978-1-84996-690-0.

BROERSMA, H. J.; LI, X.; LI, X.; WOEGINGER, G.; ZHANG, S. Paths and cycles in colored graphs. **Australasian journal of combinatorics**, v. 31, n. 1, p. 299–311, 2005.

CERULLI, R.; FINK, A.; GENTILI, M.; RAICONI, A. The k-labeled spanning forest problem. **Procedia - Social and Behavioral Sciences**, v. 108, p. 153 – 163, 2014. ISSN 1877-0428.

CHANG, R.-S.; SHING-JIUAN, L. The minimum labeling spanning trees. **Information Processing Letters**, v. 63, n. 5, p. 277 – 282, 1997. ISSN 0020-0190.

CONSOLI, S.; PéREZ, J. A. M. Variable neighbourhood search for the k-labelled spanning forest problem. **Electronic Notes in Discrete Mathematics**, v. 47, p. 29 – 36, 2015. ISSN 1571-0653. The 3rd International Conference on Variable Neighborhood Search (VNS'14).

CONSOLI, S.; PéREZ, J. A. M.; MLADENOVIć, N. Comparison of metaheuristics for the k-labeled spanning forest problem. **International Transactions in Operational Research**, v. 24, n. 3, p. 559–582, 2017.

CONSOLI, S.; PéREZ, J. M.; MLADENOVIć, N. Intelligent variable neighbourhood search for the minimum labelling spanning tree problem. **Electronic Notes in Discrete Mathematics**, v. 41, p. 399 – 406, 2013. ISSN 1571-0653.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to Algorithms, Third Edition**. 3rd. ed. Cambridge, Massachusetts,USA: The MIT Press, 2009. ISBN 0262033844, 9780262033848.

DANTZIG, G. **Linear programming and extensions**. Princeton, NJ: Princeton Univ. Press, 1963. (Rand Corporation Research Study).

GRANATA, D.; CERULLI, R.; SCUTELLÀ, M. G.; RAICONI, A. Maximum flow problems and an np-complete variant on edge-labeled graphs. In: PARDALOS, P. M.; DU, D.-Z.; GRAHAM, R. L. (Ed.). **Handbook of Combinatorial Optimization**. New York, NY: Springer New York, 2013. p. 1913–1948. ISBN 978-1-4419-7997-1.

KLEE, V.; MINTY, G. J. How good is the simplex algorithm? In: SHISHA, O. (Ed.). **Inequalities III (Proceedings of the Third Symposium on Inequalities)**. New York, NY: Academic Press, 1972. p. 159–175.

KRUMKE, S. O.; WIRTH, H.-C. On the minimum label spanning tree problem. **Information Processing Letters**, v. 66, n. 2, p. 81 – 85, 1998. ISSN 0020-0190.

MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. Integer programming formulation of traveling salesman problems. **J. ACM**, Association for Computing Machinery, New York, NY, USA, v. 7, n. 4, p. 326–329, out. 1960. ISSN 0004-5411.

PADBERG, M. W.; WOLSEY, L. A. Trees and cuts. In: BERGE, C.; BRESSON, D.; CAMION, P.; MAURRAS, J.; STERBOUL, F. (Ed.). **Combinatorial Mathematics**. Amsterdam, Netherlands: North-Holland, 1983, (North-Holland Mathematics Studies, v. 75). p. 511 – 517.

SILVA, T. G. d. **The Minimum Labeling Spanning Tree and Related Problems**. Tese (Doutorado) — Universidade Federal Fluminense, Université d'Avignon et des Pays de Vaucluse, Nitéroi, 2018.

SILVA, T. G. d.; GUEYE, S.; MICHELON, P.; OCHI, L. S.; CABRAL, L. d. A. F. A polyhedral approach to the generalized minimum labeling spanning tree problem. **EURO Journal on Computational Optimization**, v. 7, n. 1, p. 47–77, Mar 2019. ISSN 2192-4414.

STOER, M.; WAGNER, F. A simple min-cut algorithm. **J. ACM**, ACM, New York, NY, USA, v. 44, n. 4, p. 585–591, jul. 1997. ISSN 0004-5411.

VOBS, S.; FINK, A.; DUIN, C. Looking ahead with the pilot method. **Annals of Operations Research**, v. 136, n. 1, p. 285–302, Apr 2005. ISSN 1572-9338.

WAN, Y.; CHEN, G.; XU, Y. A note on the minimum label spanning tree. **Information Processing Letters**, v. 84, n. 2, p. 99 – 101, 2002. ISSN 0020-0190.

WIART, J.; WATANABE, S.; WU, T.; JOSEPH, W.; LEE, K. A. Electromagnetic fields (emf) exposure. **Annals of Telecommunications**, v. 74, n. 1, p. 1–3, Feb 2019. ISSN 1958-9395.

XIONG, Y.; GOLDEN, B.; WASIL, E. Worst-case behavior of the mvca heuristic for the minimum labeling spanning tree problem. **Operations Research Letters**, v. 33, n. 1, p. 77 – 80, 2005. ISSN 0167-6377.

APÊNDICE A – TESTES

Aqui estão alguns dos testes feitos. Nos testes para algoritmos enumerativos somente serão reportadas as soluções ótimas. Quando um modelo reportar estado de solução "desconhecido" significa que o modelo não conseguiu encontrar uma solução viável de valor igual ou melhor que *cutoff* fornecido, para as configurações de modelo usadas, dentro do tempo limite.

Tabela 3 – Comparativo entre implementações iniciais modelos

	CCut			Fluxo			Exato Serial	
instancia	tempo (s)	obj	estado	tempo (s)	obj	estado	tempo (s)	obj
100-990-25-1-7	11.9	46	Ótimo	9.77	46	Ótimo	0.016375	46
100-990-25-3-10	0.52	3	Ótimo	14.78	3	Ótimo	0.672677	3
100-990-25-3-1	0.56	3	Ótimo	28.09	3	Ótimo	0.71897	3
100-990-25-3-2	0.46	3	Ótimo	11.39	3	Ótimo	0.69564	3
100-990-25-3-3	0.11	1	Ótimo	3.16	1	Ótimo	0.360394	1
100-990-25-3-4	0.48	2	Ótimo	7.93	2	Ótimo	0.642663	2
100-990-25-3-5	0.6	3	Ótimo	16.18	3	Ótimo	0.712291	3
100-990-25-3-6	0.51	2	Ótimo	12.09	2	Ótimo	0.678963	2
100-990-25-3-8	0.32	2	Ótimo	9.6	2	Ótimo	0.735739	2
100-990-25-3-9	0.56	3	Ótimo	16.62	3	Ótimo	0.776261	3
100-990-50-3-1	2.12	16	Ótimo	5607.85	16	Ótimo	0.20054	16
100-990-50-3-2	1.94	20	Ótimo	343.24	20	Ótimo	0.386094	20
100-990-50-3-4	2.16	21	Ótimo	1633.42	21	Ótimo	0.655563	21
100-990-50-3-5	1.49	17	Ótimo	228.91	17	Ótimo	0.565972	17
100-990-50-3-6	1.82	17	Ótimo	119.59	17	Ótimo	0.343146	17
100-990-50-3-7	3.31	22	Ótimo	3643.45	22	Ótimo	0.577111	22
100-990-50-3-9	0.99	16	Ótimo	895.33	16	Ótimo	0.235395	16
100-990-50-6-10	0.16	1	Ótimo	8.17	1	Ótimo	2.47935	1
100-990-50-6-3	0.32	1	Ótimo	6.17	1	Ótimo	1.59236	1
100-990-50-6-8	0.33	1	Ótimo	9.82	1	Ótimo	0.320047	1
100-990-100-6-10	2.12	12	Ótimo	6044.94	12	Ótimo	7.94187	12
100-990-100-6-1	3.36	14	Ótimo	7264.61	15	Viável	18.4335	14
100-990-100-6-2	2.11	13	Ótimo	7264.52	14	Viável	4.85068	13
100-990-100-6-3	4.13	14	Ótimo	7261.86	14	Viável	5.89952	14
100-990-100-6-4	1.51	11	Ótimo	5471.08	11	Ótimo	6.59069	11
100-990-100-6-5	2.23	12	Ótimo	7260.01	12	Viável	4.58014	12
100-990-100-6-6	3.26	13	Ótimo	7262.53	13	Viável	7.73651	13
100-990-100-6-7	2.8	13	Ótimo	7262.99	13	Viável	1.85023	13
100-990-100-6-8	1.46	12	Ótimo	3857.07	12	Ótimo	3.74322	12
100-990-100-6-9	2.4	13	Ótimo	7261.57	13	Viável	6.24341	13
100-990-125-6-10	4.04	24	Ótimo	7261.6	24	Viável	0.52945	24
100-990-125-6-1	2.18	19	Ótimo	7267.63	21	Viável	0.227245	19
100-990-125-6-2	2.93	22	Ótimo	7269.78	24	Viável	0.313748	22
100-990-125-6-3	4.99	18	Ótimo	7260	18	Viável	1.09291	18
100-990-125-6-4	4.75	20	Ótimo	7264.76	20	Viável	1.55375	20
100-990-125-6-5	2.56	18	Ótimo	7260.71	18	Viável	0.204712	18
100-990-125-6-6	4.02	20	Ótimo	7260.01	20	Viável	0.302207	20
100-990-125-6-7	1.93	22	Ótimo	7260.01	24	Viável	0.520996	22
100-990-125-6-8	3.57	21	Ótimo	7262.15	22	Viável	1.82684	21

Continua próxima pagina

Tabela 3 – continuação pagina anterior

	CCut			Fluxo			Exato Serial	
instancia	tempo (s)	obj	estado	tempo (s)	obj	estados	tempo (s)	obj
100-990-125-6-9	3.5	21	Ótimo	7260.94	21	Viável	1.63592	21
200-3980-100-6-10	592.76	2	Ótimo	7260.01	2	Viável	-	-
200-3980-100-6-1	512.87	2	Ótimo	7260.01	2	Viável	-	-
200-3980-100-6-2	46.29	1	Ótimo	7260.01	2	Viável	83.75	1
200-3980-100-6-3	181.41	1	Ótimo	7260.02	3	Viável	881829	1
200-3980-100-6-4	181.32	1	Ótimo	7260.02	3	Viável	91.2124	1
200-3980-100-6-5	865.01	2	Ótimo	7260.01	3	Viável	-	-
200-3980-100-6-6	966.41	2	Ótimo	7260.01	2	Viável	-	-
200-3980-100-6-7	18.68	1	Ótimo	1261.68	1	Ótimo	7.62591	1
200-3980-100-6-8	1095.19	2	Ótimo	7260.01	2	Viável	-	-
200-3980-100-6-9	232.6	1	Ótimo	7260.01	2	Viável	528482	1
200-3980-200-6-10	198.56	29	Ótimo	7260.01	32	Viável	443444	29
200-3980-200-6-1	131.48	29	Ótimo	7260.02	37	Viável	572563	29
200-3980-200-6-2	115.66	28	Ótimo	7260.02	37	Viável	177	28
200-3980-200-6-3	91.16	40	Ótimo	7261.19	48	Viável	1273.09	40
200-3980-200-6-4	372.74	32	Ótimo	7260.01	46	Viável	1716.89	32
200-3980-200-6-5	546.56	34	Ótimo	7260.02	41	Viável	1985.61	34
200-3980-200-6-6	227.99	34	Ótimo	7260.02	46	Viável	1565.82	34
200-3980-200-6-7	228.94	30	Ótimo	7260.02	43	Viável	606328	30
200-3980-200-6-8	52.1	29	Ótimo	7260.01	42	Viável	456486	29
200-3980-200-6-9	618.76	37	Ótimo	7260.01	42	Viável	3509.36	37
200-3980-250-12-1	7200	2	Viável	7260.02	2	Viável	-	-
200-3980-250-12-2	761.13	1	Ótimo	7260.01	2	Viável	453337	1
200-3980-250-12-5	6091.8	1	Ótimo	6832.41	1	Ótimo	-	-
200-3980-250-12-7	7200	2	Viável	2748.78	1	Ótimo	-	-
200-3980-250-12-8	6300.75	1	Ótimo	7260.04	1	Viável	-	-
200-3980-250-6-10	48.71	45	Ótimo	7260.02	49	Viável	86.9998	45
200-3980-250-6-3	35.7	47	Ótimo	7260.02	52	Viável	4.82424	47
200-3980-250-6-4	29.14	50	Ótimo	7260.02	61	Viável	20.3716	50
200-3980-250-6-6	48.38	48	Ótimo	7260.02	58	Viável	37.0994	48
200-3980-250-6-9	67.38	49	Ótimo	7260.02	52	Viável	106749	49
200-3980-50-3-10	11.21	7	Ótimo	1053.53	7	Ótimo	18.2943	7
200-3980-50-3-1	9.67	7	Ótimo	525	7	Ótimo	18.1368	7
200-3980-50-3-2	5.94	3	Ótimo	320.23	3	Ótimo	17.6078	3
200-3980-50-3-3	5.82	7	Ótimo	1830.03	7	Ótimo	18.2602	7
200-3980-50-3-4	6.67	6	Ótimo	1157.53	6	Ótimo	17.8857	6
200-3980-50-3-5	9.2	7	Ótimo	894.9	7	Ótimo	18.1872	7
200-3980-50-3-6	6.72	6	Ótimo	1298.38	6	Ótimo	18.1717	6
200-3980-50-3-7	8.05	8	Ótimo	1100.2	8	Ótimo	17.9167	8
200-3980-50-3-8	6.93	6	Ótimo	676.21	6	Ótimo	18.2436	6
200-3980-50-3-9	6.27	6	Ótimo	3372.81	6	Ótimo	18.1107	6
300-8970-150-4-10	297.11	31	Ótimo	7260.03	47	Viável	-	-
300-8970-150-4-1	115.96	30	Ótimo	7260.04	45	Viável	6697.8	30
300-8970-150-4-2	304.83	37	Ótimo	7260.55	52	Viável	-	-
300-8970-150-4-3	299.74	34	Ótimo	7260.03	45	Viável	-	-
300-8970-150-4-4	247.67	33	Ótimo	7260.03	42	Viável	-	-
300-8970-150-4-5	333.86	32	Ótimo	7260.03	42	Viável	-	-
300-8970-150-4-6	478.33	36	Ótimo	7260.03	39	Viável	-	-
300-8970-150-4-7	265.99	38	Ótimo	7260.03	47	Viável	-	-
300-8970-150-4-8	336.78	35	Ótimo	7260.03	40	Viável	-	-

Continua próxima pagina

Tabela 3 – continuação pagina anterior

	CCut			Fluxo			Exato Serial	
instancia	tempo (s)	obj	estado	tempo (s)	obj	estados	tempo (s)	obj
300-8970-150-4-9	1120.6	40	Ótimo	7260.03	47	Viável	-	-
300-8970-300-9-10	7200	16	Viável	7260.02	29	Viável	-	-
300-8970-300-9-1	7200	13	Viável	7260.04	19	Viável	-	-
300-8970-300-9-2	7200	15	Viável	7260.03	32	Viável	-	-
300-8970-300-9-3	7200	18	Viável	7260.02	28	Viável	-	-
300-8970-300-9-4	7200	14	Viável	7262.27	31	Viável	-	-
300-8970-300-9-5	7200	19	Viável	7260.03	20	Viável	-	-
300-8970-300-9-6	7200	14	Viável	7260.03	20	Viável	-	-
300-8970-300-9-7	7200	14	Viável	7260.02	24	Viável	-	-
300-8970-300-9-8	7200	16	Viável	7260.03	31	Viável	-	-
300-8970-300-9-9	7200	13	Viável	7260.03	31	Viável	-	-
300-8970-375-9-10	7200	26	Viável	7260.03	43	Viável	-	-
300-8970-375-9-1	7200	28	Viável	7260.03	47	Viável	-	-
300-8970-375-9-2	7200	29	Viável	7260.04	60	Viável	-	-
300-8970-375-9-3	7200	30	Viável	7260.04	59	Viável	-	-
300-8970-375-9-4	7200	29	Viável	7260.05	45	Viável	-	-
300-8970-375-9-5	7200	30	Viável	7260.04	43	Viável	-	-
300-8970-375-9-6	7200	32	Viável	7260.02	76	Viável	-	-
300-8970-375-9-7	7200	31	Viável	7260.03	52	Viável	-	-
300-8970-375-9-8	7200	27	Viável	7260.03	35	Viável	-	-
300-8970-375-9-9	7200	29	Viável	7260.03	41	Viável	-	-
300-8970-75-2-3	102.85	47	Ótimo	7260.02	48	Viável	6.83072	47
300-8970-75-2-9	91.23	45	Ótimo	7260.34	48	Viável	6.9352	45
300-8970-75-4-10	39.05	1	Ótimo	7260.02	2	Viável	95.4449	1
300-8970-75-4-1	21.41	1	Ótimo	4857.04	1	Ótimo	173847	1
300-8970-75-4-2	21.33	1	Ótimo	7260.02	2	Viável	145655	1
300-8970-75-4-4	31.89	1	Ótimo	7260.03	2	Viável	8.72639	1
300-8970-75-4-5	5.81	1	Ótimo	600.68	1	Ótimo	20.4882	1
300-8970-75-4-6	13.87	1	Ótimo	7260.02	2	Viável	161178	1
300-8970-75-4-7	12.15	1	Ótimo	7260.03	2	Viável	286997	1
300-8970-75-4-8	5.33	1	Ótimo	2857.25	1	Ótimo	210386	1
400-15960-100-3-10	186.76	14	Ótimo	7260.04	24	Viável	696599	14
400-15960-100-3-1	219.64	18	Ótimo	7260.05	19	Viável	701631	18
400-15960-100-3-2	256.9	15	Ótimo	7260.05	18	Viável	683302	15
400-15960-100-3-3	336.21	19	Ótimo	7260.04	23	Viável	678632	19
400-15960-100-3-4	238.31	20	Ótimo	7260.06	24	Viável	692577	20
400-15960-100-3-5	130.3	14	Ótimo	7260.04	19	Viável	697311	14
400-15960-100-3-6	245.87	17	Ótimo	7260.06	21	Viável	672.84	17
400-15960-100-3-7	280.65	18	Ótimo	7260.04	20	Viável	702395	18
400-15960-100-3-8	318.13	16	Ótimo	7260.04	18	Viável	694616	16
400-15960-100-3-9	321.02	16	Ótimo	7260.04	22	Viável	661116	16
400-15960-200-6-10	7200	10	Viável	7260.04	18	Viável	-	-
400-15960-200-6-1	7200	8	Viável	7260.04	22	Viável	-	-
400-15960-200-6-2	7200	8	Viável	7260.05	23	Viável	-	-
400-15960-200-6-3	7200	6	Viável	7260.04	19	Viável	-	-
400-15960-200-6-4	7200	7	Viável	7260.04	28	Viável	-	-
400-15960-200-6-5	7200	7	Viável	7260.06	26	Viável	-	-
400-15960-200-6-6	7200	8	Viável	7260.04	18	Viável	-	-
400-15960-200-6-7	7200	7	Viável	7260.46	21	Viável	-	-
400-15960-200-6-8	7200	7	Viável	7260.04	16	Viável	-	-
400-15960-200-6-9	7200	7	Viável	7260.04	21	Viável	-	-
400-15960-400-12-10	7200	5	Viável	7260.05	10	Viável	-	-
400-15960-400-12-1	7200	14	Viável	7261.31	15	Viável	-	-
400-15960-400-12-2	7200	5	Viável	7260.05	7	Viável	-	-
400-15960-400-12-3	7200	5	Viável	7260.04	15	Viável	-	-
400-15960-400-12-4	7200	7	Viável	7260.04	11	Viável	-	-

Continua próxima pagina

Tabela 3 – continuação pagina anterior

	CCut			Fluxo			Exato Serial	
instancia	tempo (s)	obj	estado	tempo (s)	obj	estados	tempo (s)	obj
400-15960-400-12-6	7200	9	Viável	7260.05	10	Viável	-	-
400-15960-400-12-7	7200	12	Viável	7260.05	14	Viável	-	-
400-15960-400-12-8	7200	13	Viável	7260.04	11	Viável	-	-
400-15960-400-12-9	7200	8	Viável	7260.04	9	Viável	-	-
400-15960-400-6-5	7200	74	Viável	7260.05	117	Viável	-	-
400-15960-500-12-10	7200	20	Viável	7260.05	32	Viável	-	-
400-15960-500-12-1	7200	15	Viável	7260.03	37	Viável	-	-
400-15960-500-12-2	7200	14	Viável	7260.05	40	Viável	-	-
400-15960-500-12-3	7200	26	Viável	7260.11	22	Viável	-	-
400-15960-500-12-4	7200	23	Viável	7260.04	50	Viável	-	-
400-15960-500-12-5	7200	18	Viável	7260.04	30	Viável	-	-
400-15960-500-12-6	7200	22	Viável	7260.05	50	Viável	-	-
400-15960-500-12-7	7200	33	Viável	7260.04	34	Viável	-	-
400-15960-500-12-8	7200	24	Viável	7260.05	47	Viável	-	-
400-15960-500-12-9	7200	33	Viável	7260.06	49	Viável	-	-
500-24950-125-3-10	1603.56	22	Ótimo	7260.09	34	Viável	2279.82	22
500-24950-125-3-1	971.76	24	Ótimo	7260.07	43	Viável	2327.19	24
500-24950-125-3-2	1591.76	23	Ótimo	7260.08	36	Viável	2272.11	23
500-24950-125-3-3	860.3	17	Ótimo	7260.07	28	Viável	2323.38	17
500-24950-125-3-4	1381.7	20	Ótimo	7260.09	46	Viável	2249.92	20
500-24950-125-3-5	1694.39	24	Ótimo	7260.08	39	Viável	2298.77	24
500-24950-125-3-6	1073.13	23	Ótimo	7260.08	45	Viável	2234.11	23
500-24950-125-3-7	1024.72	23	Ótimo	7260.07	31	Viável	2300.76	23
500-24950-125-3-8	1280.94	24	Ótimo	7260.08	40	Viável	2265.62	24
500-24950-125-3-9	1029.38	20	Ótimo	7260.08	30	Viável	2313.31	20
500-24950-250-7-10	7200	5	Viável	7260.05	13	Viável	-	-
500-24950-250-7-1	7200	6	Viável	7260.1	11	Viável	-	-
500-24950-250-7-2	7200	6	Viável	7260.1	23	Viável	-	-
500-24950-250-7-3	7200	5	Viável	7260.1	15	Viável	-	-
500-24950-250-7-4	7200	5	Viável	7260.08	18	Viável	-	-
500-24950-250-7-5	7200	5	Viável	7260.07	15	Viável	-	-
500-24950-250-7-6	7200	5	Viável	7260.12	21	Viável	-	-
500-24950-250-7-7	7200	5	Viável	7260.09	24	Viável	-	-
500-24950-250-7-8	7200	3	Viável	7260.07	19	Viável	-	-
500-24950-250-7-9	7200	5	Viável	7260.08	19	Viável	-	-
500-24950-500-7-10	7200	111	Viável	7260.08	154	Viável	-	-
500-24950-500-7-1	7200	81	Viável	7260.09	350	Viável	-	-
500-24950-500-7-2	7200	81	Viável	7260.09	500	Viável	-	-
500-24950-500-7-3	7200	87	Viável	7260.08	173	Viável	-	-
500-24950-500-7-4	7200	10	Viável	7260.1	148	Viável	-	-
500-24950-500-7-5	7200	96	Viável	7260.07	134	Viável	-	-
500-24950-500-7-6	7200	80	Viável	7260.11	500	Viável	-	-
500-24950-500-7-7	7200	93	Viável	7281.43	500	Viável	-	-
500-24950-500-7-8	7200	115	Viável	7260.08	137	Viável	-	-
500-24950-500-7-9	7200	83	Viável	7260.07	150	Viável	-	-
500-24950-625-15-10	7200	23	Viável	7260.08	32	Viável	-	-
500-24950-625-15-2	7200	16	Viável	7260.1	41	Viável	-	-
500-24950-625-15-3	7200	21	Viável	7260.09	39	Viável	-	-
500-24950-625-15-4	7200	13	Viável	7260.1	500	Viável	-	-
500-24950-625-15-5	7200	23	Viável	7260.08	500	Viável	-	-
500-24950-625-15-8	7200	15	Viável	7260.1	35	Viável	-	-
500-24950-625-15-9	7200	22	Viável	7260.1	25	Viável	-	-
500-24950-625-7-1	7200	-	Desconhecido	7260.08	238	Viável	-	-
500-24950-625-7-6	7200	-	Desconhecido	7260.11	500	Viável	-	-
500-24950-625-7-7	7200	126	Viável	7260.09	377	Viável	-	-
1000-99900-250-3-10	7200	58	Viável	7260.16	1000	Viável	-	-
1000-99900-250-3-1	7200	60	Viável	7260.23	1000	Viável	-	-
1000-99900-250-3-2	7200	57	Viável	7260.25	1000	Viável	-	-
1000-99900-250-3-3	7200	52	Viável	7260.27	1000	Viável	-	-

Continua próxima pagina

Tabela 3 – continuação pagina anterior

	CCut			Fluxo			Exato Serial	
instancia	tempo (s)	obj	estado	tempo (s)	obj	estados	tempo (s)	obj
1000-99900-250-3-4	7200	61	Viável	7260.21	1000	Viável	-	-
1000-99900-250-3-5	7200	58	Viável	7260.27	1000	Viável	-	-
1000-99900-250-3-6	7200	61	Viável	7260.3	1000	Viável	-	-
1000-99900-250-3-7	7200	60	Viável	7260.24	1000	Viável	-	-
1000-99900-250-3-8	7200	51	Viável	7260.17	1000	Viável	-	-
1000-99900-250-3-9	7200	59	Viável	7260.1	1000	Viável	-	-
1000-99900-500-7-10	7200	26	Viável	7260.21	60	Viável	-	-
1000-99900-500-7-1	7200	230	Viável	7260.13	1000	Viável	-	-
1000-99900-500-7-2	7200	34	Viável	7260.19	1000	Viável	-	-
1000-99900-500-7-3	7200	30	Viável	7260.34	1000	Viável	-	-
1000-99900-500-7-4	7200	25	Viável	7260.23	75	Viável	-	-
1000-99900-500-7-5	7200	27	Viável	7260.2	1000	Viável	-	-
1000-99900-500-7-6	7200	24	Viável	7260.18	49	Viável	-	-
1000-99900-500-7-7	7200	29	Viável	7260.24	1000	Viável	-	-
1000-99900-500-7-8	7200	27	Viável	7260.13	60	Viável	-	-
1000-99900-500-7-9	7200	29	Viável	7260.3	1000	Viável	-	-
1000-99900-1000-15-1	7200	35	Viável	7260.37	1000	Viável	-	-
1000-99900-1000-15-2	7200	25	Viável	7260.46	1000	Viável	-	-
1000-99900-1000-15-3	7200	24	Viável	7260.06	1000	Viável	-	-
1000-99900-1000-15-7	7200	25	Viável	7260.39	47	Viável	-	-
1000-99900-1000-15-8	7200	23	Viável	7260.29	1000	Viável	-	-
1000-99900-1000-15-9	7200	13	Viável	7260.51	41	Viável	-	-
1000-99900-1000-7-10	7200	-	Desconhecido	7260.17	1000	Viável	-	-
1000-99900-1000-7-4	7200	251	Viável	7260.27	301	Viável	-	-
1000-99900-1000-7-5	7200	261	Viável	7260.32	1000	Viável	-	-
1000-99900-1000-7-6	7200	-	Desconhecido	7260.38	1000	Viável	-	-
1000-99900-1250-15-10	7200	65	Viável	7260.53	1000	Viável	-	-
1000-99900-1250-15-1	7200	65	Viável	7260.5	1000	Viável	-	-
1000-99900-1250-15-2	7200	60	Viável	7260.17	1000	Viável	-	-
1000-99900-1250-15-3	7200	72	Viável	7260.22	1000	Viável	-	-
1000-99900-1250-15-4	7200	66	Viável	7260.44	1000	Viável	-	-
1000-99900-1250-15-5	7200	47	Viável	7260.31	1000	Viável	-	-
1000-99900-1250-15-6	7200	60	Viável	7260.44	1000	Viável	-	-
1000-99900-1250-15-7	7200	79	Viável	7260.28	1000	Viável	-	-
1000-99900-1250-15-8	7200	72	Viável	7260.25	117	Viável	-	-
1000-99900-1250-15-9	7200	73	Viável	7260.44	1000	Viável	-	-

Tabela 4 – Testes com algoritmo *backtracking* serial e paralelo

	Exato Paralelo		Exato Serial	
instancia	tempo(s)	obj	tempo (s)	obj
100-990-25-1-7	0.0010222	46	0.016375	46
100-990-25-3-10	0.0203945	3	0.672677	3
100-990-25-3-1	0.0229692	3	0.71897	3
100-990-25-3-2	0.0193987	3	0.69564	3
100-990-25-3-3	0.0144375	1	0.360394	1
100-990-25-3-4	0.017302	2	0.642663	2
100-990-25-3-5	0.02101	3	0.712291	3
100-990-25-3-6	0.0197885	2	0.678963	2
100-990-25-3-8	0.0198157	2	0.735739	2
100-990-25-3-9	0.0204896	3	0.776261	3
100-990-50-3-1	0.0313925	16	0.20054	16
100-990-50-3-2	0.0282676	20	0.386094	20
100-990-50-3-4	0.0273465	21	0.655563	21

Continua próxima pagina

Tabela 4 – continuação pagina anterior

	Exato Paralelo		Exato Serial	
instancia	tempo(s)	obj	tempo (s)	obj
100-990-50-3-5	0.030176	17	0.565972	17
100-990-50-3-6	0.0334411	17	0.343146	17
100-990-50-3-7	0.0277044	22	0.577111	22
100-990-50-3-9	0.0287507	16	0.235395	16
100-990-50-6-10	1.42447	1	2.47935	1
100-990-50-6-3	0.380936	1	1.59236	1
100-990-50-6-8	0.290973	1	0.320047	1
100-990-100-6-10	0.458949	12	7.94187	12
100-990-100-6-1	0.515793	14	18.4335	14
100-990-100-6-2	0.458051	13	4.85068	13
100-990-100-6-3	0.53157	14	5.89952	14
100-990-100-6-4	0.436446	11	6.59069	11
100-990-100-6-5	0.457004	12	4.58014	12
100-990-100-6-6	0.596067	13	7.73651	13
100-990-100-6-7	0.563545	13	1.85023	13
100-990-100-6-8	0.452853	12	3.74322	12
100-990-100-6-9	0.794873	13	6.24341	13
100-990-125-6-10	0.361594	24	0.52945	24
100-990-125-6-1	0.60621	19	0.227245	19
100-990-125-6-2	0.37704	22	0.313748	22
100-990-125-6-3	0.454087	18	1.09291	18
100-990-125-6-4	0.203697	20	1.55375	20
100-990-125-6-5	0.325223	18	0.204712	18
100-990-125-6-6	0.40758	20	0.302207	20
100-990-125-6-7	0.298629	22	0.520996	22
100-990-125-6-8	0.433556	21	1.82684	21
100-990-125-6-9	0.391563	21	1.63592	21
200-3980-100-6-10	1208.1	2	-	-
200-3980-100-6-1	933.151	2	-	-
200-3980-100-6-2	79.4357	1	83.75	1
200-3980-100-6-3	72.965	1	881.829	1
200-3980-100-6-4	41.15	1	91.2124	1
200-3980-100-6-5	1273.08	2	-	-
200-3980-100-6-6	1239.69	2	-	-
200-3980-100-6-7	6.79044	1	7.62591	1
200-3980-100-6-8	1226.7	2	-	-
200-3980-100-6-9	107.102	1	528.482	1
200-3980-200-6-10	7.19034	29	443.444	29
200-3980-200-6-1	6.70605	29	572.563	29
200-3980-200-6-2	10.3154	28	177	28
200-3980-200-6-3	5.67665	40	1273.09	40
200-3980-200-6-4	11.0223	32	1716.89	32
200-3980-200-6-5	11.7877	34	1985.61	34
200-3980-200-6-6	6.78569	34	1565.82	34
200-3980-200-6-7	11.9545	30	606.328	30
200-3980-200-6-8	3.95336	29	456.486	29
200-3980-200-6-9	10.8242	37	3509.36	37
200-3980-250-12-1	-	-	-	-
200-3980-250-12-2	-	-	453.337	1
200-3980-250-12-5	-	-	-	-
200-3980-250-12-7	-	-	-	-
200-3980-250-12-8	-	-	-	-
200-3980-250-6-10	3.50483	45	86.9998	45
200-3980-250-6-3	3.45242	47	4.82424	47
200-3980-250-6-4	4.20431	50	20.3716	50
200-3980-250-6-6	3.47391	48	37.0994	48
200-3980-250-6-9	2.38469	49	106.749	49
200-3980-50-3-10	0.257494	7	18.2943	7
200-3980-50-3-1	0.249613	7	18.1368	7
200-3980-50-3-2	0.232745	3	17.6078	3
200-3980-50-3-3	0.250669	7	18.2602	7

Continua próxima pagina

Tabela 4 – continuação pagina anterior

	Exato Paralelo		Exato Serial	
instancia	tempo(s)	obj	tempo (s)	obj
200-3980-50-3-4	0.254442	6	17.8857	6
200-3980-50-3-5	0.246831	7	18.1872	7
200-3980-50-3-6	0.234271	6	18.1717	6
200-3980-50-3-7	0.267094	8	17.9167	8
200-3980-50-3-8	0.233026	6	18.2436	6
200-3980-50-3-9	0.250339	6	18.1107	6
300-8970-150-4-10	12.5205	31	-	-
300-8970-150-4-1	9.32296	30	6697.8	30
300-8970-150-4-2	15.2768	35	-	-
300-8970-150-4-3	15.0487	34	-	-
300-8970-150-4-4	12.2513	33	-	-
300-8970-150-4-5	13.8382	32	-	-
300-8970-150-4-6	20.1543	36	-	-
300-8970-150-4-7	16.1636	38	-	-
300-8970-150-4-8	16.6712	35	-	-
300-8970-150-4-9	25.0248	40	-	-
300-8970-300-9-10	-	-	-	-
300-8970-300-9-1	-	-	-	-
300-8970-300-9-2	-	-	-	-
300-8970-300-9-3	-	-	-	-
300-8970-300-9-4	-	-	-	-
300-8970-300-9-5	-	-	-	-
300-8970-300-9-6	-	-	-	-
300-8970-300-9-7	-	-	-	-
300-8970-300-9-8	-	-	-	-
300-8970-300-9-9	-	-	-	-
300-8970-375-9-10	-	-	-	-
300-8970-375-9-1	-	-	-	-
300-8970-375-9-2	-	-	-	-
300-8970-375-9-3	-	-	-	-
300-8970-375-9-4	-	-	-	-
300-8970-375-9-5	-	-	-	-
300-8970-375-9-6	-	-	-	-
300-8970-375-9-7	-	-	-	-
300-8970-375-9-8	-	-	-	-
300-8970-375-9-9	-	-	-	-
300-8970-75-2-3	0.0931994	47	6.83072	47
300-8970-75-2-9	0.062036	45	6.9352	45
300-8970-75-4-10	4.10572	1	95.4449	1
300-8970-75-4-1	7.64139	1	173.847	1
300-8970-75-4-2	8.01549	1	145.655	1
300-8970-75-4-4	0.55529	1	8.72639	1
300-8970-75-4-5	0.833153	1	20.4882	1
300-8970-75-4-6	4.8107	1	161.178	1
300-8970-75-4-7	4.04198	1	286.997	1
300-8970-75-4-8	1.095	1	210.386	1
400-15960-100-3-10	5.85986	14	696.599	14
400-15960-100-3-1	5.76508	18	701.631	18
400-15960-100-3-2	5.90545	15	683.302	15
400-15960-100-3-3	5.90323	19	678.632	19
400-15960-100-3-4	5.94848	20	692.577	20
400-15960-100-3-5	5.83868	14	697.311	14
400-15960-100-3-6	5.75835	17	672.84	17
400-15960-100-3-7	5.75647	18	702.395	18
400-15960-100-3-8	5.98348	16	694.616	16
400-15960-100-3-9	5.89622	16	661.116	16
400-15960-200-6-10	-	-	-	-
400-15960-200-6-1	-	-	-	-
400-15960-200-6-2	-	-	-	-
400-15960-200-6-3	-	-	-	-
400-15960-200-6-4	-	-	-	-

Continua próxima pagina

Tabela 4 – continuação pagina anterior

	Exato Paralelo		Exato Serial	
instancia	tempo(s)	obj	tempo (s)	obj
400-15960-200-6-5	-	-	-	-
400-15960-200-6-6	-	-	-	-
400-15960-200-6-7	-	-	-	-
400-15960-200-6-8	-	-	-	-
400-15960-200-6-9	-	-	-	-
400-15960-400-12-10	-	-	-	-
400-15960-400-12-1	-	-	-	-
400-15960-400-12-2	-	-	-	-
400-15960-400-12-3	-	-	-	-
400-15960-400-12-4	-	-	-	-
400-15960-400-12-6	-	-	-	-
400-15960-400-12-7	-	-	-	-
400-15960-400-12-8	-	-	-	-
400-15960-400-12-9	-	-	-	-
400-15960-400-6-5	395.018	74	-	-
400-15960-500-12-10	-	-	-	-
400-15960-500-12-1	-	-	-	-
400-15960-500-12-2	-	-	-	-
400-15960-500-12-3	-	-	-	-
400-15960-500-12-4	-	-	-	-
400-15960-500-12-5	-	-	-	-
400-15960-500-12-6	-	-	-	-
400-15960-500-12-7	-	-	-	-
400-15960-500-12-8	-	-	-	-
400-15960-500-12-9	-	-	-	-
500-24950-125-3-10	15.2044	22	2279.82	22
500-24950-125-3-1	15.9046	24	2327.19	24
500-24950-125-3-2	15.4424	23	2272.11	23
500-24950-125-3-3	15.6498	17	2323.38	17
500-24950-125-3-4	15.7347	20	2249.92	20
500-24950-125-3-5	15.0475	24	2298.77	24
500-24950-125-3-6	15.7089	23	2234.11	23
500-24950-125-3-7	15.8836	23	2300.76	23
500-24950-125-3-8	15.8872	24	2265.62	24
500-24950-125-3-9	15.1589	20	2313.31	20
500-24950-250-7-10	-	-	-	-
500-24950-250-7-1	-	-	-	-
500-24950-250-7-2	-	-	-	-
500-24950-250-7-3	-	-	-	-
500-24950-250-7-4	-	-	-	-
500-24950-250-7-5	-	-	-	-
500-24950-250-7-6	-	-	-	-
500-24950-250-7-7	-	-	-	-
500-24950-250-7-8	-	-	-	-
500-24950-250-7-9	-	-	-	-
500-24950-500-7-10	-	-	-	-
500-24950-500-7-1	-	-	-	-
500-24950-500-7-2	-	-	-	-
500-24950-500-7-3	-	-	-	-
500-24950-500-7-4	-	-	-	-
500-24950-500-7-5	-	-	-	-
500-24950-500-7-6	-	-	-	-
500-24950-500-7-7	-	-	-	-
500-24950-500-7-8	-	-	-	-
500-24950-500-7-9	-	-	-	-
500-24950-625-15-10	-	-	-	-
500-24950-625-15-2	-	-	-	-
500-24950-625-15-3	-	-	-	-
500-24950-625-15-4	-	-	-	-
500-24950-625-15-5	-	-	-	-
500-24950-625-15-8	-	-	-	-

Continua próxima pagina

Tabela 4 – continuação pagina anterior

	Exato Paralelo		Exato Serial	
instancia	tempo(s)	obj	tempo (s)	obj
500-24950-625-15-9	-	-	-	-
500-24950-625-7-1	1073.47	111	-	-
500-24950-625-7-6	2460	108	-	-
500-24950-625-7-7	818.172	119	-	-
1000-99900-250-3-10	462.342	55	-	-
1000-99900-250-3-1	496.667	57	-	-
1000-99900-250-3-2	641.378	57	-	-
1000-99900-250-3-3	477.142	52	-	-
1000-99900-250-3-4	469.356	57	-	-
1000-99900-250-3-5	551.895	58	-	-
1000-99900-250-3-6	465.116	59	-	-
1000-99900-250-3-7	634.006	60	-	-
1000-99900-250-3-8	533.685	51	-	-
1000-99900-250-3-9	567.924	51	-	-
1000-99900-500-7-10	-	-	-	-
1000-99900-500-7-1	-	-	-	-
1000-99900-500-7-2	-	-	-	-
1000-99900-500-7-3	-	-	-	-
1000-99900-500-7-4	-	-	-	-
1000-99900-500-7-5	-	-	-	-
1000-99900-500-7-6	-	-	-	-
1000-99900-500-7-7	-	-	-	-
1000-99900-500-7-8	-	-	-	-
1000-99900-500-7-9	-	-	-	-
1000-99900-1000-15-1	-	-	-	-
1000-99900-1000-15-2	-	-	-	-
1000-99900-1000-15-3	-	-	-	-
1000-99900-1000-15-7	-	-	-	-
1000-99900-1000-15-8	-	-	-	-
1000-99900-1000-15-9	-	-	-	-
1000-99900-1000-7-10	-	-	-	-
1000-99900-1000-7-4	-	-	-	-
1000-99900-1000-7-5	-	-	-	-
1000-99900-1000-7-6	-	-	-	-
1000-99900-1250-15-10	-	-	-	-
1000-99900-1250-15-1	-	-	-	-
1000-99900-1250-15-2	-	-	-	-
1000-99900-1250-15-3	-	-	-	-
1000-99900-1250-15-4	-	-	-	-
1000-99900-1250-15-5	-	-	-	-
1000-99900-1250-15-6	-	-	-	-
1000-99900-1250-15-7	-	-	-	-
1000-99900-1250-15-8	-	-	-	-
1000-99900-1250-15-9	-	-	-	-

Tabela 5 – Comparativo entre *B&C*-Ccut 1 e 2

	B&C-Ccut2			B&C-Ccut1		
instância	tempo (s)	obj	estado	tempo (s)	obj	estado
100-990-25-1-7	0	46	Ótimo	11.9	46	Ótimo
100-990-25-3-10	0.07	3	Ótimo	0.52	3	Ótimo
100-990-25-3-1	0.08	3	Ótimo	0.56	3	Ótimo
100-990-25-3-2	0.06	3	Ótimo	0.46	3	Ótimo
100-990-25-3-3	0.03	1	Ótimo	0.11	1	Ótimo

Continua próxima pagina

Tabela 5 – continuação pagina anterior

	B&C-Ccut2			B&C-Ccut1		
instância	tempo (s)	obj	estado	tempo (s)	obj	estado
100-990-25-3-4	0.05	2	Ótimo	0.48	2	Ótimo
100-990-25-3-5	0.1	3	Ótimo	0.6	3	Ótimo
100-990-25-3-6	0.06	2	Ótimo	0.51	2	Ótimo
100-990-25-3-8	0.05	2	Ótimo	0.32	2	Ótimo
100-990-25-3-9	0.09	3	Ótimo	0.56	3	Ótimo
100-990-50-3-1	0.02	16	Ótimo	2.12	16	Ótimo
100-990-50-3-2	0.03	20	Ótimo	1.94	20	Ótimo
100-990-50-3-4	0.04	21	Ótimo	2.16	21	Ótimo
100-990-50-3-5	0.05	17	Ótimo	1.49	17	Ótimo
100-990-50-3-6	0.08	17	Ótimo	1.82	17	Ótimo
100-990-50-3-7	0.07	22	Ótimo	3.31	22	Ótimo
100-990-50-3-9	0.03	16	Ótimo	0.99	16	Ótimo
100-990-50-6-10	0.01	1	Ótimo	0.16	1	Ótimo
100-990-50-6-3	0.07	1	Ótimo	0.32	1	Ótimo
100-990-50-6-8	0.07	1	Ótimo	0.33	1	Ótimo
100-990-100-6-10	0.72	12	Ótimo	2.12	12	Ótimo
100-990-100-6-1	1.36	14	Ótimo	3.36	14	Ótimo
100-990-100-6-2	1.18	13	Ótimo	2.11	13	Ótimo
100-990-100-6-3	0.84	14	Ótimo	4.13	14	Ótimo
100-990-100-6-4	0.8	11	Ótimo	1.51	11	Ótimo
100-990-100-6-5	0.75	12	Ótimo	2.23	12	Ótimo
100-990-100-6-6	0.65	13	Ótimo	3.26	13	Ótimo
100-990-100-6-7	0.38	13	Ótimo	2.8	13	Ótimo
100-990-100-6-8	0.58	12	Ótimo	1.46	12	Ótimo
100-990-100-6-9	0.92	13	Ótimo	2.4	13	Ótimo
100-990-125-6-10	0.18	24	Ótimo	4.04	24	Ótimo
100-990-125-6-1	0.02	19	Ótimo	2.18	19	Ótimo
100-990-125-6-2	0.08	22	Ótimo	2.93	22	Ótimo
100-990-125-6-3	0.86	18	Ótimo	4.99	18	Ótimo
100-990-125-6-4	0.37	20	Ótimo	4.75	20	Ótimo
100-990-125-6-5	0.13	18	Ótimo	2.56	18	Ótimo
100-990-125-6-6	0.13	20	Ótimo	4.02	20	Ótimo
100-990-125-6-7	0.04	22	Ótimo	1.93	22	Ótimo
100-990-125-6-8	0.4	21	Ótimo	3.57	21	Ótimo
100-990-125-6-9	0.51	21	Ótimo	3.5	21	Ótimo
200-3980-100-6-10	54.19	2	Ótimo	592.76	2	Ótimo
200-3980-100-6-1	59.93	2	Ótimo	512.87	2	Ótimo
200-3980-100-6-2	10.15	1	Ótimo	46.29	1	Ótimo
200-3980-100-6-3	36.11	1	Ótimo	181.41	1	Ótimo
200-3980-100-6-4	28.49	1	Ótimo	181.32	1	Ótimo
200-3980-100-6-5	115.8	2	Ótimo	865.01	2	Ótimo
200-3980-100-6-6	91.04	2	Ótimo	966.41	2	Ótimo
200-3980-100-6-7	0.67	1	Ótimo	18.68	1	Ótimo
200-3980-100-6-8	83.25	2	Ótimo	1095.19	2	Ótimo
200-3980-100-6-9	54.98	1	Ótimo	232.6	1	Ótimo
200-3980-200-6-10	10.5	29	Ótimo	198.56	29	Ótimo
200-3980-200-6-1	8.6	29	Ótimo	131.48	29	Ótimo
200-3980-200-6-2	3.28	28	Ótimo	115.66	28	Ótimo
200-3980-200-6-3	3.7	40	Ótimo	91.16	40	Ótimo

Continua próxima pagina

Tabela 5 – continuação pagina anterior

	B&C-Ccut2			B&C-Ccut1		
instância	tempo (s)	obj	estado	tempo (s)	obj	estado
200-3980-200-6-4	13.54	32	Ótimo	372.74	32	Ótimo
200-3980-200-6-5	13.42	34	Ótimo	546.56	34	Ótimo
200-3980-200-6-6	12.03	34	Ótimo	227.99	34	Ótimo
200-3980-200-6-7	8.89	30	Ótimo	228.94	30	Ótimo
200-3980-200-6-8	11.34	29	Ótimo	52.1	29	Ótimo
200-3980-200-6-9	28.49	37	Ótimo	618.76	37	Ótimo
200-3980-250-12-1	13.37	1	Ótimo	7200	2	Viável
200-3980-250-12-2	2.37	1	Ótimo	761.13	1	Ótimo
200-3980-250-12-5	2855.41	1	Ótimo	6091.8	1	Ótimo
200-3980-250-12-7	22.16	1	Ótimo	7200	2	Viável
200-3980-250-12-8	95.91	1	Ótimo	6300.75	1	Ótimo
200-3980-250-6-10	0.42	45	Ótimo	48.71	45	Ótimo
200-3980-250-6-3	0.21	47	Ótimo	35.7	47	Ótimo
200-3980-250-6-4	0.32	50	Ótimo	29.14	50	Ótimo
200-3980-250-6-6	0.99	48	Ótimo	48.38	48	Ótimo
200-3980-250-6-9	1.98	49	Ótimo	67.38	49	Ótimo
200-3980-50-3-10	1.58	7	Ótimo	11.21	7	Ótimo
200-3980-50-3-1	1.43	7	Ótimo	9.67	7	Ótimo
200-3980-50-3-2	0.56	3	Ótimo	5.94	3	Ótimo
200-3980-50-3-3	1.41	7	Ótimo	5.82	7	Ótimo
200-3980-50-3-4	1.29	6	Ótimo	6.67	6	Ótimo
200-3980-50-3-5	1.45	7	Ótimo	9.2	7	Ótimo
200-3980-50-3-6	1.22	6	Ótimo	6.72	6	Ótimo
200-3980-50-3-7	1.49	8	Ótimo	8.05	8	Ótimo
200-3980-50-3-8	1.21	6	Ótimo	6.93	6	Ótimo
200-3980-50-3-9	1.38	6	Ótimo	6.27	6	Ótimo
300-8970-150-4-10	23.99	31	Ótimo	297.11	31	Ótimo
300-8970-150-4-1	34.63	30	Ótimo	115.96	30	Ótimo
300-8970-150-4-2	59.42	35	Ótimo	304.83	37	Ótimo
300-8970-150-4-3	41.04	34	Ótimo	299.74	34	Ótimo
300-8970-150-4-4	32.04	33	Ótimo	247.67	33	Ótimo
300-8970-150-4-5	31.77	32	Ótimo	333.86	32	Ótimo
300-8970-150-4-6	68.55	36	Ótimo	478.33	36	Ótimo
300-8970-150-4-7	43.07	38	Ótimo	265.99	38	Ótimo
300-8970-150-4-8	37.85	35	Ótimo	336.78	35	Ótimo
300-8970-150-4-9	133.69	40	Ótimo	1120.6	40	Ótimo
300-8970-300-9-10	7200	-	Desconhecido	7200	16	Viável
300-8970-300-9-1	7200	-	Desconhecido	7200	13	Viável
300-8970-300-9-2	7200	-	Desconhecido	7200	15	Viável
300-8970-300-9-3	7200	-	Desconhecido	7200	18	Viável
300-8970-300-9-4	7200	-	Desconhecido	7200	14	Viável
300-8970-300-9-5	7200	11	Viável	7200	19	Viável
300-8970-300-9-6	7200	-	Desconhecido	7200	14	Viável
300-8970-300-9-7	7200	-	Desconhecido	7200	14	Viável
300-8970-300-9-8	7200	-	Desconhecido	7200	16	Viável
300-8970-300-9-9	7200	-	Desconhecido	7200	13	Viável
300-8970-375-9-10	7200	-	Desconhecido	7200	26	Viável
300-8970-375-9-1	7200	-	Desconhecido	7200	28	Viável
300-8970-375-9-2	7200	-	Desconhecido	7200	29	Viável
300-8970-375-9-3	7200	-	Desconhecido	7200	30	Viável
300-8970-375-9-4	7200	-	Desconhecido	7200	29	Viável
300-8970-375-9-5	7200	-	Desconhecido	7200	30	Viável
300-8970-375-9-6	7200	-	Desconhecido	7200	32	Viável

Continua próxima pagina

Tabela 5 – continuação pagina anterior

	B&C-Ccut2			B&C-Ccut1		
instância	tempo (s)	obj	estado	tempo (s)	obj	estado
300-8970-375-9-7	7200	-	Desconhecido	7200	31	Viável
300-8970-375-9-8	7200	-	Desconhecido	7200	27	Viável
300-8970-375-9-9	7200	-	Desconhecido	7200	29	Viável
300-8970-75-2-3	1.9	47	Ótimo	102.85	47	Ótimo
300-8970-75-2-9	0.6	45	Ótimo	91.23	45	Ótimo
300-8970-75-4-10	13.96	1	Ótimo	39.05	1	Ótimo
300-8970-75-4-1	11.33	1	Ótimo	21.41	1	Ótimo
300-8970-75-4-2	3.11	1	Ótimo	21.33	1	Ótimo
300-8970-75-4-4	1.5	1	Ótimo	31.89	1	Ótimo
300-8970-75-4-5	2.02	1	Ótimo	5.81	1	Ótimo
300-8970-75-4-6	1.75	1	Ótimo	13.87	1	Ótimo
300-8970-75-4-7	0.19	1	Ótimo	12.15	1	Ótimo
300-8970-75-4-8	0.51	1	Ótimo	5.33	1	Ótimo
400-15960-100-3-10	19.81	14	Ótimo	186.76	14	Ótimo
400-15960-100-3-1	23.99	18	Ótimo	219.64	18	Ótimo
400-15960-100-3-2	21.41	15	Ótimo	256.9	15	Ótimo
400-15960-100-3-3	25.19	19	Ótimo	336.21	19	Ótimo
400-15960-100-3-4	27.11	20	Ótimo	238.31	20	Ótimo
400-15960-100-3-5	21.64	14	Ótimo	130.3	14	Ótimo
400-15960-100-3-6	24.43	17	Ótimo	245.87	17	Ótimo
400-15960-100-3-7	23.86	18	Ótimo	280.65	18	Ótimo
400-15960-100-3-8	23.97	16	Ótimo	318.13	16	Ótimo
400-15960-100-3-9	23.35	16	Ótimo	321.02	16	Ótimo
400-15960-200-6-10	7200	10	Viável	7200	10	Viável
400-15960-200-6-1	7200	8	Viável	7200	8	Viável
400-15960-200-6-2	7200	8	Viável	7200	8	Viável
400-15960-200-6-3	7200	-	Desconhecido	7200	6	Viável
400-15960-200-6-4	7200	7	Viável	7200	7	Viável
400-15960-200-6-5	7200	8	Viável	7200	7	Viável
400-15960-200-6-6	7200	7	Viável	7200	8	Viável
400-15960-200-6-7	7200	9	Viável	7200	7	Viável
400-15960-200-6-8	7200	9	Viável	7200	7	Viável
400-15960-200-6-9	7200	6	Viável	7200	7	Viável
400-15960-400-12-10	7200	-	Desconhecido	7200	5	Viável
400-15960-400-12-1	7200	-	Desconhecido	7200	14	Viável
400-15960-400-12-2	7200	-	Desconhecido	7200	5	Viável
400-15960-400-12-3	7200	2	Viável	7200	5	Viável
400-15960-400-12-4	7200	-	Desconhecido	7200	7	Viável
400-15960-400-12-6	7200	-	Desconhecido	7200	9	Viável
400-15960-400-12-7	7200	-	Desconhecido	7200	12	Viável
400-15960-400-12-8	7200		Desconhecido	7200	13	Viável
400-15960-400-12-9	7200	-	Desconhecido	7200	8	Viável
400-15960-400-6-5	202.28	74	Ótimo	7200	74	Viável
400-15960-500-12-10	7200	-	Desconhecido	7200	20	Viável
400-15960-500-12-1	7200	-	Desconhecido	7200	15	Viável
400-15960-500-12-2	7200	-	Desconhecido	7200	14	Viável
400-15960-500-12-3	7200	-	Desconhecido	7200	26	Viável
400-15960-500-12-4	7200	-	Desconhecido	7200	23	Viável
400-15960-500-12-5	7200	-	Desconhecido	7200	18	Viável
400-15960-500-12-6	7200	-	Desconhecido	7200	22	Viável
400-15960-500-12-7	7200	-	Desconhecido	7200	33	Viável
400-15960-500-12-8	7200	-	Desconhecido	7200	24	Viável
400-15960-500-12-9	7200	-	Desconhecido	7200	33	Viável
500-24950-125-3-10	69.56	22	Ótimo	1603.56	22	Ótimo
500-24950-125-3-1	72.02	24	Ótimo	971.76	24	Ótimo

Continua próxima pagina

Tabela 5 – continuação pagina anterior

	B&C-Ccut2			B&C-Ccut1		
instância	tempo (s)	obj	estado	tempo (s)	obj	estado
500-24950-125-3-2	65.79	23	Ótimo	1591.76	23	Ótimo
500-24950-125-3-3	60.79	17	Ótimo	860.3	17	Ótimo
500-24950-125-3-4	65.03	20	Ótimo	1381.7	20	Ótimo
500-24950-125-3-5	70.98	24	Ótimo	1694.39	24	Ótimo
500-24950-125-3-6	67.47	23	Ótimo	1073.13	23	Ótimo
500-24950-125-3-7	65.06	23	Ótimo	1024.72	23	Ótimo
500-24950-125-3-8	77.16	24	Ótimo	1280.94	24	Ótimo
500-24950-125-3-9	68.26	20	Ótimo	1029.38	20	Ótimo
500-24950-250-7-10	7200	-	Desconhecido	7200	5	Viável
500-24950-250-7-1	7200	-	Desconhecido	7200	6	Viável
500-24950-250-7-2	7200	-	Desconhecido	7200	6	Viável
500-24950-250-7-3	7200	3	Viável	7200	5	Viável
500-24950-250-7-4	7200	-	Desconhecido	7200	5	Viável
500-24950-250-7-5	7200	5	Viável	7200	5	Viável
500-24950-250-7-6	7200	5	Viável	7200	5	Viável
500-24950-250-7-7	7200	-	Desconhecido	7200	5	Viável
500-24950-250-7-8	7200	-	Desconhecido	7200	3	Viável
500-24950-250-7-9	7200	-	Desconhecido	7200	5	Viável
500-24950-500-7-10	7200	-	Desconhecido	7200	111	Viável
500-24950-500-7-1	7200	70	Viável	7200	810	Viável
500-24950-500-7-2	7200	-	Desconhecido	7200	81	Viável
500-24950-500-7-3	7200	72	Viável	7200	87	Viável
500-24950-500-7-4	7200	-	Desconhecido	7200	10	Viável
500-24950-500-7-5	7200	-	Desconhecido	7200	960	Viável
500-24950-500-7-6	7200		Desconhecido	7200	80	Viável
500-24950-500-7-7	7200	75	Viável	7200	93	Viável
500-24950-500-7-8	7200	-	Desconhecido	7200	115	Viável
500-24950-500-7-9	7200	72	Viável	7200	83	Viável
500-24950-625-15-10	7200	3	Viável	7200	23	Viável
500-24950-625-15-2	7200	-	Desconhecido	7200	16	Viável
500-24950-625-15-3	7200	-	Desconhecido	7200	210	Viável
500-24950-625-15-4	7200	-	Desconhecido	7200	13	Viável
500-24950-625-15-5	7200	-	Desconhecido	7200	23	Viável
500-24950-625-15-8	7200	-	Desconhecido	7200	15	Viável
500-24950-625-15-9	7200	-	Desconhecido	7200	22	Viável
500-24950-625-7-1	440.77	111	Ótimo	7200	-	Desconhecido
500-24950-625-7-6	4310.84	108	Ótimo	7200	-	Desconhecido
500-24950-625-7-7	1151.7	119	Ótimo	7200	126	Viável
1000-99900-250-3-10	7200	-	Desconhecido	7200	58	Viável
1000-99900-250-3-1	7200	-	Desconhecido	7200	60	Viável
1000-99900-250-3-2	7200	-	Desconhecido	7200	57	Viável
1000-99900-250-3-3	7200	-	Desconhecido	7200	52	Viável
1000-99900-250-3-4	7200	-	Desconhecido	7200	61	Viável
1000-99900-250-3-5	7200	-	Desconhecido	7200	58	Viável
1000-99900-250-3-6	7200	-	Desconhecido	7200	61	Viável
1000-99900-250-3-7	7200	-	Desconhecido	7200	60	Viável
1000-99900-250-3-8	1558.01	51	Ótimo	7200	51	Viável
1000-99900-250-3-9	1305.59	51	Ótimo	7200	59	Viável
1000-99900-500-7-10	7200	-	Desconhecido	7200	26	Viável
1000-99900-500-7-1	7200	-	Desconhecido	7200	23	Viável
1000-99900-500-7-2	7200	-	Desconhecido	7200	34	Viável
1000-99900-500-7-3	7200	-	Desconhecido	7200	30	Viável
1000-99900-500-7-4	7200	-	Desconhecido	7200	25	Viável
1000-99900-500-7-5	7200	-	Desconhecido	7200	27	Viável
1000-99900-500-7-6	7200	-	Desconhecido	7200	24	Viável
1000-99900-500-7-7	7200	-	Desconhecido	7200	29	Viável
1000-99900-500-7-8	7200	-	Desconhecido	7200	27	Viável
1000-99900-500-7-9	7200	-	Desconhecido	7200	29	Viável

Continua próxima pagina

Tabela 5 – continuação pagina anterior

	B&C-Ccut2			B&C-Ccut1		
instância	tempo (s)	obj	estado	tempo (s)	obj	estado
1000-99900-1000-15-1	7200	-	Desconhecido	7200	35	Viável
1000-99900-1000-15-2	7200	-	Desconhecido	7200	25	Viável
1000-99900-1000-15-3	7200	-	Desconhecido	7200	24	Viável
1000-99900-1000-15-7	7200	-	Desconhecido	7200	25	Viável
1000-99900-1000-15-8	7200	-	Desconhecido	7200	23	Viável
1000-99900-1000-15-9	7200	-	Desconhecido	7200	13	Viável
1000-99900-1000-7-10	7200	-	Desconhecido	7200	-	Desconhecido
1000-99900-1000-7-4	7200	-	Desconhecido	7200	251	Viável
1000-99900-1000-7-5	7200	183	Viável	7200	261	Viável
1000-99900-1000-7-6	7200	191	Viável	7200	-	Desconhecido
1000-99900-1250-15-10	7200	-	Desconhecido	7200	65	Viável
1000-99900-1250-15-1	7200	-	Desconhecido	7200	65	Viável
1000-99900-1250-15-2	7200	-	Desconhecido	7200	60	Viável
1000-99900-1250-15-3	7200	-	Desconhecido	7200	72	Viável
1000-99900-1250-15-4	7200	-	Desconhecido	7200	66	Viável
1000-99900-1250-15-5	7200	-	Desconhecido	7200	47	Viável
1000-99900-1250-15-6	7200	-	Desconhecido	7200	60	Viável
1000-99900-1250-15-7	7200	-	Desconhecido	7200	79	Viável
1000-99900-1250-15-8	7200	-	Desconhecido	7200	72	Viável
1000-99900-1250-15-9	7200	-	Desconhecido	7200	73	Viável

Tabela 6 – Comparativo - modelo fluxo clássico vs decomposição por Benders

	Benders			Fluxo Normal		
instancia	tempo(s)	obj	estado	tempo (s)	obj	estado
100-990-25-1-7	0.28	46	Ótimo	9.77	46	Ótimo
100-990-25-3-10	0.26	3	Ótimo	14.78	3	Ótimo
100-990-25-3-1	0.22	3	Ótimo	28.09	3	Ótimo
100-990-25-3-2	0.19	3	Ótimo	11.39	3	Ótimo
100-990-25-3-3	0.08	1	Ótimo	3.16	1	Ótimo
100-990-25-3-4	0.16	2	Ótimo	7.93	2	Ótimo
100-990-25-3-5	0.22	3	Ótimo	16.18	3	Ótimo
100-990-25-3-6	0.25	2	Ótimo	12.09	2	Ótimo
100-990-25-3-8	0.16	2	Ótimo	9.6	2	Ótimo
100-990-25-3-9	0.21	3	Ótimo	16.62	3	Ótimo
100-990-50-3-1	0.32	16	Ótimo	5607.85	16	Ótimo
100-990-50-3-2	0.84	20	Ótimo	343.24	20	Ótimo
100-990-50-3-4	1.37	21	Ótimo	1633.42	21	Ótimo
100-990-50-3-5	0.57	17	Ótimo	228.91	17	Ótimo
100-990-50-3-6	0.33	17	Ótimo	119.59	17	Ótimo
100-990-50-3-7	2.14	22	Ótimo	3643.45	22	Ótimo
100-990-50-3-9	0.52	16	Ótimo	895.33	16	Ótimo
100-990-50-6-10	0.16	1	Ótimo	8.17	1	Ótimo
100-990-50-6-3	0.18	1	Ótimo	6.17	1	Ótimo
100-990-50-6-8	0.13	1	Ótimo	9.82	1	Ótimo
100-990-100-6-10	223.06	12	Ótimo	6044.94	12	Ótimo
100-990-100-6-1	761.33	14	Ótimo	7264.61	15	Viável
100-990-100-6-2	387.8	13	Ótimo	7264.52	14	Viável
100-990-100-6-3	780.81	14	Ótimo	7261.86	14	Viável

Continua próxima pagina

Tabela 6 – continuação pagina anterior

	Benders			Fluxo Normal		
instancia	tempo(s)	obj	estado	tempo (s)	obj	estado
100-990-100-6-4	112.77	11	Ótimo	5471.08	11	Ótimo
100-990-100-6-5	340.51	12	Ótimo	7260.01	12	Viável
100-990-100-6-6	2230.36	13	Ótimo	7262.53	13	Viável
100-990-100-6-7	493.99	13	Ótimo	7262.99	13	Viável
100-990-100-6-8	235.74	12	Ótimo	3857.07	12	Ótimo
100-990-100-6-9	201.03	13	Ótimo	7261.57	13	Viável
100-990-125-6-10	7300.06	24	Viável	7261.6	24	Viável
100-990-125-6-1	178.27	19	Ótimo	7267.63	21	Viável
100-990-125-6-2	7300.07	22	Viável	7269.78	24	Viável
100-990-125-6-3	51.4	18	Ótimo	7260	18	Viável
100-990-125-6-4	1853.48	20	Ótimo	7264.76	20	Viável
100-990-125-6-5	326.05	18	Ótimo	7260.71	18	Viável
100-990-125-6-6	3061.1	20	Ótimo	7260.01	20	Viável
100-990-125-6-7	7300.1	22	Viável	7260.01	24	Viável
100-990-125-6-8	884.85	21	Ótimo	7262.15	22	Viável
100-990-125-6-9	1507.87	21	Ótimo	7260.94	21	Viável
200-3980-100-6-10	398.09	2	Ótimo	7260.01	2	Viável
200-3980-100-6-1	150.54	2	Ótimo	7260.01	2	Viável
200-3980-100-6-2	2.36	1	Ótimo	7260.01	2	Viável
200-3980-100-6-3	16.76	1	Ótimo	7260.02	3	Viável
200-3980-100-6-4	9.43	1	Ótimo	7260.02	3	Viável
200-3980-100-6-5	474.53	2	Ótimo	7260.01	3	Viável
200-3980-100-6-6	467.95	2	Ótimo	7260.01	2	Viável
200-3980-100-6-7	0.59	1	Ótimo	1261.68	1	Ótimo
200-3980-100-6-8	1029.83	2	Ótimo	7260.01	2	Viável
200-3980-100-6-9	4.87	1	Ótimo	7260.01	2	Viável
200-3980-200-6-10	7300.01	-	Desconhecido	7260.01	32	Viável
200-3980-200-6-1	7300.03	-	Desconhecido	7260.02	37	Viável
200-3980-200-6-2	7300.01	-	Desconhecido	7260.02	37	Viável
200-3980-200-6-3	7300.01	-	Desconhecido	7261.19	48	Viável
200-3980-200-6-4	7300.01	-	Desconhecido	7260.01	46	Viável
200-3980-200-6-5	7300.01	-	Desconhecido	7260.02	41	Viável
200-3980-200-6-6	7300.01	-	Desconhecido	7260.02	46	Viável
200-3980-200-6-7	7300.02	-	Desconhecido	7260.02	43	Viável
200-3980-200-6-8	7300.01	30	Viável	7260.01	42	Viável
200-3980-200-6-9	7300.03	-	Desconhecido	7260.01	42	Viável
200-3980-250-12-1	7300.02	-	Desconhecido	7260.02	2	Viável
200-3980-250-12-2	160.37	1	Ótimo	7260.01	2	Viável
200-3980-250-12-5	1.45	1	Ótimo	6832.41	1	Ótimo
200-3980-250-12-7	2.28	1	Ótimo	2748.78	1	Ótimo
200-3980-250-12-8	0.73	1	Ótimo	7260.04	1	Viável
200-3980-250-6-10	7300.06	-	Desconhecido	7260.02	49	Viável
200-3980-250-6-3	7300.05	-	Desconhecido	7260.02	52	Viável
200-3980-250-6-4	7502.33	-	Desconhecido	7260.02	61	Viável
200-3980-250-6-6	7502.33	-	Desconhecido	7260.02	58	Viável
200-3980-250-6-9	7502.33	-	Desconhecido	7260.02	52	Viável
200-3980-50-3-10	2.94	7	Ótimo	1053.53	7	Ótimo
200-3980-50-3-1	3.35	7	Ótimo	525	7	Ótimo
200-3980-50-3-2	1.37	3	Ótimo	320.23	3	Ótimo
200-3980-50-3-3	2.94	7	Ótimo	1830.03	7	Ótimo
200-3980-50-3-4	1.63	6	Ótimo	1157.53	6	Ótimo
200-3980-50-3-5	3.26	7	Ótimo	894.9	7	Ótimo
200-3980-50-3-6	1.04	6	Ótimo	1298.38	6	Ótimo

Continua próxima pagina

Tabela 6 – continuação pagina anterior

	Benders			Fluxo Normal		
instancia	tempo(s)	obj	estado	tempo (s)	obj	estado
200-3980-50-3-7	3.23	8	Ótimo	1100.2	8	Ótimo
200-3980-50-3-8	1.48	6	Ótimo	676.21	6	Ótimo
200-3980-50-3-9	2.1	6	Ótimo	3372.81	6	Ótimo
300-8970-150-4-10	4668.78	31	Ótimo	7260.03	47	Viável
300-8970-150-4-1	1953.76	30	Ótimo	7260.04	45	Viável
300-8970-150-4-2	5022.1	35	Ótimo	7260.55	52	Viável
300-8970-150-4-3	7500.01	34	Viável	7260.03	45	Viável
300-8970-150-4-4	2772.9	33	Ótimo	7260.03	42	Viável
300-8970-150-4-5	2486.67	32	Ótimo	7260.03	42	Viável
300-8970-150-4-6	7500.01	37	Viável	7260.03	39	Viável
300-8970-150-4-7	7500.02	38	Viável	7260.03	47	Viável
300-8970-150-4-8	6996.07	35	Ótimo	7260.03	40	Viável
300-8970-150-4-9	7500.01	42	Viável	7260.03	47	Viável
300-8970-300-9-10	7504.66	-	Desconhecido	7260.02	29	Viável
300-8970-300-9-1	7502.35	-	Desconhecido	7260.04	19	Viável
300-8970-300-9-2	7502.99	-	Desconhecido	7260.03	32	Viável
300-8970-300-9-3	7506.11	-	Desconhecido	7260.02	28	Viável
300-8970-300-9-4	7505.19	-	Desconhecido	7262.27	31	Viável
300-8970-300-9-5	7503.11	-	Desconhecido	7260.03	20	Viável
300-8970-300-9-6	7500.01	11	Viável	7260.03	20	Viável
300-8970-300-9-7	7502.73	-	Desconhecido	7260.02	24	Viável
300-8970-300-9-8	7503.08	-	Desconhecido	7260.03	31	Viável
300-8970-300-9-9	7500.01	-	Desconhecido	7260.03	31	Viável
300-8970-375-9-10	7511.14	-	Desconhecido	7260.03	43	Viável
300-8970-375-9-1	7506.37	-	Desconhecido	7260.03	47	Viável
300-8970-375-9-2	7512.44	-	Desconhecido	7260.04	60	Viável
300-8970-375-9-3	7515.05	-	Desconhecido	7260.04	59	Viável
300-8970-375-9-4	7505.98	-	Desconhecido	7260.05	45	Viável
300-8970-375-9-5	7505.07	-	Desconhecido	7260.04	43	Viável
300-8970-375-9-6	7505.76	-	Desconhecido	7260.02	76	Viável
300-8970-375-9-7	7508.84	-	Desconhecido	7260.03	52	Viável
300-8970-375-9-8	7513.22	-	Desconhecido	7260.03	35	Viável
300-8970-375-9-9	7511.63	-	Desconhecido	7260.03	41	Viável
300-8970-75-2-3	13.2	47	Ótimo	7260.02	48	Viável
300-8970-75-2-9	13.64	45	Ótimo	7260.34	48	Viável
300-8970-75-4-10	1.42	1	Ótimo	7260.02	2	Viável
300-8970-75-4-1	5.34	1	Ótimo	4857.04	1	Ótimo
300-8970-75-4-2	1.39	1	Ótimo	7260.02	2	Viável
300-8970-75-4-4	4.23	1	Ótimo	7260.03	2	Viável
300-8970-75-4-5	2.28	1	Ótimo	600.68	1	Ótimo
300-8970-75-4-6	1.36	1	Ótimo	7260.02	2	Viável
300-8970-75-4-7	5.08	1	Ótimo	7260.03	2	Viável
300-8970-75-4-8	1.7	1	Ótimo	2857.25	1	Ótimo
400-15960-100-3-10	26.62	14	Ótimo	7260.04	24	Viável
400-15960-100-3-1	107.82	18	Ótimo	7260.05	19	Viável
400-15960-100-3-2	38.81	15	Ótimo	7260.05	18	Viável
400-15960-100-3-3	55.91	19	Ótimo	7260.04	23	Viável
400-15960-100-3-4	132.39	20	Ótimo	7260.06	24	Viável
400-15960-100-3-5	61.74	14	Ótimo	7260.04	19	Viável
400-15960-100-3-6	27.97	17	Ótimo	7260.06	21	Viável
400-15960-100-3-7	36.76	18	Ótimo	7260.04	20	Viável
400-15960-100-3-8	100.27	16	Ótimo	7260.04	18	Viável
400-15960-100-3-9	35.85	16	Ótimo	7260.04	22	Viável
400-15960-200-6-10	7504.08	8	Viável	7260.04	18	Viável

Continua próxima pagina

Tabela 6 – continuação pagina anterior

	Benders			Fluxo Normal		
instancia	tempo(s)	obj	estado	tempo (s)	obj	estado
400-15960-200-6-1	7504.7	7	Viável	7260.04	22	Viável
400-15960-200-6-2	7513.47	7	Viável	7260.05	23	Viável
400-15960-200-6-3	7504.82	6	Viável	7260.04	19	Viável
400-15960-200-6-4	7503.44	8	Viável	7260.04	28	Viável
400-15960-200-6-5	7502.63	8	Viável	7260.06	26	Viável
400-15960-200-6-6	7505.78	7	Viável	7260.04	18	Viável
400-15960-200-6-7	7509.62	8	Viável	7260.46	21	Viável
400-15960-200-6-8	7502.84	7	Viável	7260.04	16	Viável
400-15960-200-6-9	7503.88	9	Viável	7260.04	21	Viável
400-15960-400-12-10	9.58	1	Ótimo	7260.05	10	Viável
400-15960-400-12-1	6972.07	1	Ótimo	7261.31	15	Viável
400-15960-400-12-2	7506.85	-	Desconhecido	7260.05	7	Viável
400-15960-400-12-3	246.12	1	Ótimo	7260.04	15	Viável
400-15960-400-12-4	500.03	1	Ótimo	7260.04	11	Viável
400-15960-400-12-6	7200	-	Desconhecido	7260.05	10	Viável
400-15960-400-12-7	7507.5	-	Desconhecido	7260.05	14	Viável
400-15960-400-12-8	6863.04	1	Ótimo	7260.04	11	Viável
400-15960-400-12-9	7515.83	-	Desconhecido	7260.04	9	Viável
400-15960-400-6-5	7508.73	-	Desconhecido	7260.05	117	Viável
400-15960-500-12-10	7508.91	-	Desconhecido	7260.05	32	Viável
400-15960-500-12-1	7504.83	-	Desconhecido	7260.03	37	Viável
400-15960-500-12-2	7506.47	-	Desconhecido	7260.05	40	Viável
400-15960-500-12-3	7510.99	-	Desconhecido	7260.11	22	Viável
400-15960-500-12-4	7505.16	-	Desconhecido	7260.04	50	Viável
400-15960-500-12-5	7508.24	-	Desconhecido	7260.04	30	Viável
400-15960-500-12-6	7507.62	-	Desconhecido	7260.05	50	Viável
400-15960-500-12-7	7506.63	-	Desconhecido	7260.04	34	Viável
400-15960-500-12-8	7513.26	-	Desconhecido	7260.05	47	Viável
400-15960-500-12-9	7505.91	-	Desconhecido	7260.06	49	Viável
500-24950-125-3-10	173.08	22	Ótimo	7260.09	34	Viável
500-24950-125-3-1	433.04	24	Ótimo	7260.07	43	Viável
500-24950-125-3-2	168.88	23	Ótimo	7260.08	36	Viável
500-24950-125-3-3	80.13	17	Ótimo	7260.07	28	Viável
500-24950-125-3-4	34.87	20	Ótimo	7260.09	46	Viável
500-24950-125-3-5	93.32	24	Ótimo	7260.08	39	Viável
500-24950-125-3-6	213.16	23	Ótimo	7260.08	45	Viável
500-24950-125-3-7	60.82	23	Ótimo	7260.07	31	Viável
500-24950-125-3-8	288.22	24	Ótimo	7260.08	40	Viável
500-24950-125-3-9	81.63	20	Ótimo	7260.08	30	Viável
500-24950-250-7-10	7512.88	4	Viável	7260.05	13	Viável
500-24950-250-7-1	7508.82	4	Viável	7260.1	11	Viável
500-24950-250-7-2	7511.82	-	Desconhecido	7260.1	23	Viável
500-24950-250-7-3	7519.49	-	Desconhecido	7260.1	15	Viável
500-24950-250-7-4	7517.43	5	Viável	7260.08	18	Viável
500-24950-250-7-5	7513.78	3	Viável	7260.07	15	Viável
500-24950-250-7-6	7503.83	-	Desconhecido	7260.12	21	Viável
500-24950-250-7-7	7524.82	5	Viável	7260.09	24	Viável
500-24950-250-7-8	7502.75	-	Desconhecido	7260.07	19	Viável
500-24950-250-7-9	7517.99	4	Viável	7260.08	19	Viável
500-24950-500-7-10	7505.42	-	Desconhecido	7260.08	154	Viável
500-24950-500-7-1	7514.02	-	Desconhecido	7260.09	350	Viável
500-24950-500-7-2	7505.29	-	Desconhecido	7260.09	500	Viável
500-24950-500-7-3	7526.54	-	Desconhecido	7260.08	173	Viável
500-24950-500-7-4	7527.19	-	Desconhecido	7260.1	148	Viável
500-24950-500-7-5	7506.85	-	Desconhecido	7260.07	134	Viável
500-24950-500-7-6	7518.3	-	Desconhecido	7260.11	500	Viável
500-24950-500-7-7	7508.61	-	Desconhecido	7281.43	500	Viável

Continua próxima pagina

Tabela 6 – continuação pagina anterior

	Benders			Fluxo Normal		
instancia	tempo(s)	obj	estado	tempo (s)	obj	estado
500-24950-500-7-8	7533.23	-	Desconhecido	7260.08	137	Viável
500-24950-500-7-9	7521.94	-	Desconhecido	7260.07	150	Viável
500-24950-625-15-10	1863.7	1	Ótimo	7260.08	32	Viável
500-24950-625-15-2	7517.67	-	Desconhecido	7260.1	41	Viável
500-24950-625-15-3	7506.91	-	Desconhecido	7260.09	39	Viável
500-24950-625-15-4	7535.51	-	Desconhecido	7260.1	500	Viável
500-24950-625-15-5	7507.13	-	Desconhecido	7260.08	500	Viável
500-24950-625-15-8	7508.54	-	Desconhecido	7260.1	35	Viável
500-24950-625-15-9	7502.33	-	Desconhecido	7260.1	25	Viável
500-24950-625-7-1	7519.26	-	Desconhecido	7260.08	238	Viável
500-24950-625-7-6	7521.01	-	Desconhecido	7260.11	500	Viável
500-24950-625-7-7	7500	-	Desconhecido	7260.09	377	Viável
1000-99900-250-3-10	7499.98	55	Viável	7260.16	1000	Viável
1000-99900-250-3-1	7499.98	57	Viável	7260.23	1000	Viável
1000-99900-250-3-2	7499.99	57	Viável	7260.25	1000	Viável
1000-99900-250-3-3	7499.99	55	Viável	7260.27	1000	Viável
1000-99900-250-3-4	7499.99	58	Viável	7260.21	1000	Viável
1000-99900-250-3-5	7499.99	60	Viável	7260.27	1000	Viável
1000-99900-250-3-6	7499.99	60	Viável	7260.3	1000	Viável
1000-99900-250-3-7	7500	60	Viável	7260.24	1000	Viável
1000-99900-250-3-8	5489.01	51	Ótimo	7260.17	1000	Viável
1000-99900-250-3-9	7499.99	51	Viável	7260.1	1000	Viável
1000-99900-500-7-10	7504.65	-	Desconhecido	7260.21	60	Viável
1000-99900-500-7-1	7503.07	-	Desconhecido	7260.13	1000	Viável
1000-99900-500-7-2	7507.38	-	Desconhecido	7260.19	1000	Viável
1000-99900-500-7-3	7507.02	-	Desconhecido	7260.34	1000	Viável
1000-99900-500-7-4	7511.93	-	Desconhecido	7260.23	75	Viável
1000-99900-500-7-5	7504.66	-	Desconhecido	7260.2	1000	Viável
1000-99900-500-7-6	7502.33	-	Desconhecido	7260.18	49	Viável
1000-99900-500-7-7	7502.33	-	Desconhecido	7260.24	1000	Viável
1000-99900-500-7-8	7502.33	-	Desconhecido	7260.13	60	Viável
1000-99900-500-7-9	7507.17	-	Desconhecido	7260.3	1000	Viável
1000-99900-1000-15-1	7503.66	-	Desconhecido	7260.37	1000	Viável
1000-99900-1000-15-2	7528.78	-	Desconhecido	7260.46	1000	Viável
1000-99900-1000-15-3	7521.08	-	Desconhecido	7260.06	1000	Viável
1000-99900-1000-15-7	7504.5	-	Desconhecido	7260.39	47	Viável
1000-99900-1000-15-8	7503.62	-	Desconhecido	7260.29	1000	Viável
1000-99900-1000-15-9	7505.5	-	Desconhecido	7260.51	41	Viável
1000-99900-1000-7-10	7510.82	-	Desconhecido	7260.17	1000	Viável
1000-99900-1000-7-4	7504.92	-	Desconhecido	7260.27	301	Viável
1000-99900-1000-7-5	7505.7	-	Desconhecido	7260.32	1000	Viável
1000-99900-1000-7-6	7503.07	-	Desconhecido	7260.38	1000	Viável
1000-99900-1250-15-10	7504.27	-	Desconhecido	7260.53	1000	Viável
1000-99900-1250-15-1	7499.98	-	Desconhecido	7260.5	1000	Viável
1000-99900-1250-15-2	7522.28	-	Desconhecido	7260.17	1000	Viável
1000-99900-1250-15-3	7511.94	-	Desconhecido	7260.22	1000	Viável
1000-99900-1250-15-4	7516.94	-	Desconhecido	7260.44	1000	Viável
1000-99900-1250-15-5	7499.98	-	Desconhecido	7260.31	1000	Viável
1000-99900-1250-15-6	7515.84	-	Desconhecido	7260.44	1000	Viável
1000-99900-1250-15-7	7506.71	-	Desconhecido	7260.28	1000	Viável
1000-99900-1250-15-8	7501.8	-	Desconhecido	7260.25	117	Viável
1000-99900-1250-15-9	7499.98	-	Desconhecido	7260.44	1000	Viável