UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
PROGRAMA DE MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

Paulo Henrique Macêdo de Araújo

Uma Heurística Lagrangeana para o Problema de Ponderação de Rodadas

FORTALEZA – CE
Fevereiro de 2014

Paulo Henrique Macêdo de Araújo

Uma Heurística Lagrangeana para o Problema de Ponderação de Rodadas

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

Orientador: Prof. Dr. Ricardo Cordeiro Corrêa

FORTALEZA – CE
Fevereiro de 2014

PAULO HENRIQUE MACÊDO DE ARAÚJO

Uma Heurística Lagrangeana com Paralelismo para o Problema de Ponderação de Rodadas

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

BANCA EXAMINADORA

Prof. Dr. Ricardo Cordeiro Corrêa
(Orientador)
Universidade Federal do Ceará – UFC

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

Prof. Dr. Plácido Rogério Pinheiro
Universidade de Fortaleza - Unifor

Fortaleza, 20 de fevereiro de 2014

Aos meus pais, Herbênia e Paulo.

Agradecimentos

Agradeço primeiramente à Deus, por estar sempre guiando meu caminho.

Ao meu orientador, professor Ricardo Cordeiro Corrêa, por ter aceitado o desafio de orientar este trabalho e ajudado imensamente no meu processo de aprendizado, não só durante o mestrado, mas também na graduação. Obrigado pela paciência, confiança e dedicação em todos os momentos.

Ao membro da banca, professor Rafael Castro de Andrade, pelas fundamentais sugestões, revisões do trabalho e incentivos durante todo o trabalho.

Ao membro da banca, professor Plácido Rogério Pinheiro, por ter aceitado participar da banca e pelos importantes comentários e motivação.

Ao professor Manoel Campêlo, que infelizmente não pôde fazer parte da banca, mas foi muito importante para que eu tivesse conseguido realizar este trabalho. Esteve sempre disposto a me ajudar quando precisei e agradeço de coração.

À Cristiana, pela grandiosa ajuda no início do trabalho.

Ao professor Valmir Carneiro Barbosa e ao Fabio Rocha Jimenez Vieira por terem me recebido durante o meu estágio na COPPE/UFRJ.

À minha mãe Herbênia e ao meu pai Paulo, exemplos de honestidade e dignidade, por terem me dado todo o suporte e educação para que eu pudesse estar aqui nesse momento.

À minha namorada, Raiza, por estar sempre ao meu lado, me apoiando nos momentos mais difíceis com toda a dedicação, atenção, compreensão e paciência.

Ao meu irmão, Alessandro, por me ajudar nos estudos e a manter sempre o bom humor.

Aos meus familiares que sempre me apoiaram, incentivaram e torceram por mim.

Aos meus amigos, Abelardo, Eliezer, Gustavo, Lucas e Rennan, pelos vários momentos divertidos e tensos que passamos.

Ao pessoal do lab2, Arthur, Cláudio, Márcio, Rafael, Samuel, Tatiane e Vinícius, pelo momentos engraçados que passamos no lab2.

Ao grupo de pesquisa ParGO, aos professores Carlos Eduardo Fish de Brito, Pablo Mayckon Silva Farias e Victor Almeida Campos e demais professores, pela ajuda e disponibilidade.

Aos amigos, funcionários e professores do MDCC/UFC.

A todas as pessoas que, direta ou indiretamente, contribuíram para a realização desta Dissertação de Mestrado.

À CAPES e ao CNPq pelo fornecimento de bolsas de estudos que garantiu o sustento financeiro necessário para a realização deste trabalho.

"O conhecimento é um tesouro, mas a prática é a chave para alcançá-lo".

(Thomas Fuller)

Resumo

Nesta dissertação, nosso principal objetivo foi desenvolver uma técnica de resolução para um problema na área de telecomunicações. O problema em questão é chamado de problema de *Ponderação de Rodadas* (*PR*) e foi inicialmente proposto em (KLASING; MORALES; PéRENNES, 2008). O contexto do problema envolve uma rede sem fio, onde as comunicações são realizadas via ondas de rádio e a rede funciona através de uma operação da rede que satisfaz certas restrições. Inicialmente, explicamos como é formada uma rede de rádio e descrevemos a forma de operação da rede de rádio junto às restrições usando um modelo matemático. Em seguida, formalizamos o problema *PR* como um problema de otimização, especificando suas restrições, correspondente à geração do conjunto de possíveis operações da rede, e critério de otimização, referente ao uso dos recursos da rede. Posteriormente, mostramos um estudo preliminar do problema de *Coloração Fracionária* (*CF*) e apresentamos uma técnica de resolução desse problema através do uso de uma heurística lagrangeana baseada em uma relaxação lagrangeana de uma formulação de programação inteira do problema. Essa técnica de resolução é então adaptada para o problema *PR*, consistindo na principal contribuição de nossa pesquisa. Por fim, mostramos os resultados computacionais e análises das nossas implementações para os problemas *CF* e *PR*.

Palavras-chave: Telecomunicações, Redes de Rádio, Problema de Ponderação de Rodadas, Formulação por Representantes, Heurística Lagrangeana, Paralelismo.

Abstract

In this dissertation, our main objective was to develop a technique for resolution to a problem in the area of telecommunications. The problem in question is called *Round Weighting Problem* (*RWP*) and was originally proposed in (KLASING; MORALES; PéRENNES, 2008). The context of the problem involves a wireless network where communications are performed by radio waves and the network operates through a network operation that satisfies the constraints of the problem. Initially, we explain how a radio network is formed and describe the mode of operation of the radio network with restrictions using a mathematical model. Then, we formalize the *RWP* as an optimization problem, specifying their restrictions, corresponding to the generation of the set of possible network operations, and optimization criterion, regarding the use of network resources. Subsequently, we show a preliminary study of the *Fractional Coloring* problem (*FC* problem) and present a technique to solve this problem through the use of a lagrangian heuristic based on a lagrangian relaxation of an integer programming formulation of the problem. This resolution technique is then adapted to the *RWP*, consisting in the main contribution of our research. Finally, we show the computational results and analyzes of our implementations for the *Fractional Coloring* problem and *RWP*.

Key-words: Telecommunications, Radio Networks, Round Weighting Problem, Formulation by Representatives, Lagrangian Heuristic, Parallelism.

Sumário

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CF	*Coloração Fracionária*
CIMP	*Conjunto Independente Máximo Ponderado*
DSATUR	*Degree SATURation*
FCP	*Fractional Coloring Procedure*
MS	*Método do Subgradiente*
MSDF	*Método do Subgradiente de Duas Fases*
PR	*Ponderação de Rodadas*
SER	*Schedule Edge by Reversal*
WFCP	*Weighted Fractional Coloring Procedure*

1 Introdução

Aplicações relevantes de problemas de coloração aparecem de diversas formas na área de telecomunicações. No âmbito das redes sem fio, nas quais as comunicações são realizadas via ondas de rádio sujeitas a interferências, duas aplicações se destacam. Uma diz respeito a comunicações ponderadas por demandas relativas nos nós (GOMES, 2009) (KLASING; MORALES; PéRENNES, 2008) e a outra a comunicações restritas a caminhos pré-definidos (VIEIRA et al., 2012). Nesta dissertação, tratamos da primeira aplicação, para o qual damos algumas contribuições quanto à resolução do mesmo quando modelado com programação linear-inteira. As principais contribuições são uma relaxação lagrangeana derivada do modelo de programação inteira, uma heurística baseada nessa relaxação e uma avaliação experimental.

No decorrer deste capítulo introdutório, descrevemos a forma de uma operação de uma rede de rádio, explicitando as suas restrições sob a forma de um modelo matemático. Nesse sentido, estabelece-se um conjunto de possíveis operações da rede segundo o cenário de demanda de comunicação. Em seguida, detalhamos os problemas mencionados acima como problemas de otimização do uso dos recursos de uma rede de rádio. Cada um desses problemas é descrito através da especificação das restrições sobre o conjunto de possíveis operações e o critério de otimização.

1.1 Modelagem de Comunicações em Redes de Rádio

Uma *Rede de Rádio de Malha* (*RRM*) é composta por quatro elementos principais, a saber: clientes, roteadores, pontes de ligação e canais de comunicação. Os três primeiros são os elementos que geram transmissões de informação. Uma *transmissão* é realizada através da emissão de um sinal de rádio da antena de um elemento, sinal que trafega através de um canal de comunicação e é recebido pela antena de um outro elemento. Os dois elementos envolvidos na transmissão são os extremos do canal de comunicação. Nessa rede, os clientes se comunicam com roteadores, que, por sua vez, se comunicam com outros roteadores e pontes de ligação de forma a providenciar o acesso dos clientes a uma rede maior, como a Internet. Dessa forma, o fluxo de informações que transita nessa rede é referente à troca de informações entre clientes e a rede maior. Nas aplicações mencionadas no início do texto, consideramos problemas de fluxo de informações entre roteadores e pontes de ligação fixos em uma *RRM*. Para o escoamento das mensagens enviadas pelos roteadores através da rede, qualquer ponte de ligação pode receber e processar informações recebidas. Isso é devido ao fato de que qualquer ponte de ligação tem acesso à rede maior.

Uma limitação à eficiência do uso da *RRM* é devido a uma combinação de dois fatores, a saber: alcance finito de cada transmissão (como resultado da potência de emissão

das antenas e das perdas causadas pelo meio físico na propagação das ondas de rádio) e interferências causadas por transmissões simultâneas (devido ao fato de as transmissões serem realizadas em uma mesma frequência). O alcance finito de cada transmissão impõe limitações sobre os canais que podem existir na rede e, consequentemente, pode ser necessária a realização de propagação de uma mensagem através de múltiplos nós na rede a fim de alcançar o destino final da mesma. Nesse caso, um nó transmite uma mensagem para outro nó que irá encaminhá-la até alcançar o destino final. As interferências, naturalmente, limitam as possibilidades de ocorrência de transmissões simultâneas na rede.

A *RRM* é modelada por um grafo, chamado de *Grafo da Rede* (denotado por $G = (V, E)$), conexo e simples, em que os vértices em V representam os nós da rede (roteadores e pontes de ligação) e as arestas em E as possíveis transmissões em canais de comunicação (o conjunto dos arcos de G é denotado por $A = \{uv, vu \mid uv \in E\}$). Nessa rede, há nós identificados como de *origem* e de *destino*, identificados pelos subconjuntos V_O e $V_D \subset V$, respectivamente. Nós de origem são nós que representam roteadores que enviam mensagens geradas por clientes a eles ligados, através da rede. Nós de destino são nós que representam pontes de ligação que processam localmente uma mensagem quando recebida. Nós de destino não repassam mensagens, a menos que ele seja também nó de origem (esta é uma situação pouco comum na prática, mas teoricamente possível).

Algumas hipóteses são feitas em nosso modelo quanto às transmissões através da *RRM*: (i) o tamanho da mensagem enviada em uma transmissão é *unitário*; (ii) o tempo consumido em uma transmissão é constante e igual a uma unidade de tempo (neste caso, supõe-se que os tempos de atraso referentes aos processamentos realizados na saída e na chegada da mensagem são desconsiderados); (iii) cada nó da rede dispõe de uma memória de retenção de dados para armazenar mensagens em trânsito (ou seja, uma mensagem recebida em uma transmissão é retida até ser retransmitida em uma transmissão posterior. A memória de retenção de um nó de destino está sempre vazia pois esse nó processa toda mensagem que chega, sem armazená-la.); (iv) as possíveis interferências definem um grafo, denominado *Grafo de Interferências* $G_I = (E, I)$ (ver ilustração na Figura 1), cujos vértices representam os arcos de G (um vértice para os dois arcos uv e vu de cada aresta $uv \in E$) e cujas arestas indicam a ocorrência de interferência em transmissões simultâneas em arcos de G (mais especificamente, se $ee' \in I$ então transmissões simultâneas em e e e' interferem-se mutuamente). A rigor, em cada vértice deveria haver um laço, indicando que duas transmissões idênticas sempre se interferem. Por simplicidade, esses laços são omitidos; (v) a *RRM* está submetida à alta carga, o que significa que os nós de origem estão continuamente gerando demanda de comunicação; (vi) a operação da rede se faz com transmissões sincronizadas, conforme a descrição na próxima seção.

1.2 Operação de uma RRM

A operação da rede segue um padrão das transmissões a serem efetuadas, levando em conta as restrições impostas pelo modelo de operação da rede. Iniciamos pela definição da notação. Uma transmissão $\vec{uv}$ é composta por uma aresta uv de G e uma orientação dessa aresta indicando a origem (u) e o destino (v) dessa transmissão. Uma *rodada* é um conjunto de transmissões que podem ocorrer simultaneamente, ou seja, $R = \{\vec{e_1}, \ldots, \vec{e_k}\}$ tal que $(e_i, e_j) \notin I$, para todo $i, j \in \{1, \ldots, |E|\}$. Neste caso, não é possível a repetição de transmissões em uma rodada, pois as mesmas se interfeririam. Um *protocolo* $\mathcal{R}$ é uma coleção de rodadas (o que permite repetições de uma mesma rodada) satisfazendo restrições que passamos a descrever. A quantidade de vezes que uma transmissão $\vec{e}$ aparece no protocolo $\mathcal{R}$ é denominado de *fluxo* em $\vec{e}$ relativo a $\mathcal{R}$, denotado por $\phi_{\mathcal{R}}(\vec{e})$ (o índice $\mathcal{R}$ pode ser omitido quando o protocolo estiver claro pelo contexto). Observe que todas as rodadas de $\mathcal{R}$ contendo $\vec{e}$ são contadas, levando em conta sua multiplicidade em $\mathcal{R}$. As restrições para que uma coleção de rodadas forme um protocolo são as seguintes:

- **Conservação de fluxo**: $\sum_{v \in N(u)}(\phi_{\mathcal{R}}(\vec{uv}) - \phi_{\mathcal{R}}(\vec{vu})) = 0$, $\forall u \in V - \{V_O \cup V_D\}$

As restrições de **Conservação de fluxo** garantem que a quantidade de fluxo que chega em um nó é a mesma quantidade que sai.

Um *escalonamento* $\vec{\mathcal{R}}$ é uma ordenação de um protocolo $\mathcal{R}$. A execução de um escalonamento consiste em efetuar simultaneamente as transmissões de cada rodada, tomando as rodadas na sequência estabelecida. Nessa execução, cada orientação $\vec{e}$ de uma aresta $e \in E$ é ativada $\phi(\vec{e})$ vezes, sendo $\phi(\vec{e})$ o número de rodadas que incluem $\vec{e}$, representando o fluxo de informações escoadas através da aresta na orientação indicada por $\vec{e}$. Para cada transmissão, a fila de saída é do tipo *FIFO*: um nó deve escolher a mensagem que está guardada a mais tempo em sua memória de retenção. No caso de um nó de origem, caso sua memória de retenção esteja vazia, envia uma mensagem gerada por ele mesmo (FARIAS, 2014). Uma operação da rede consiste em execuções repetidas de um escalonamento. Nessa operação, o *período*, π, é o tempo gasto para uma execução do escalonamento. Como cada rodada tem suas transmissões executadas simultaneamente em uma unidade de tempo, π equivale então ao número de rodadas do protocolo.

Dado um escalonamento $\vec{\mathcal{R}}$ sendo executado repetidamente, o saldo de *fluxo* saindo de um nó de origem u em um período é dado por $\sum_{v \in N(u)}(\phi_{\vec{\mathcal{R}}}(\vec{uv}) - \phi_{\vec{\mathcal{R}}}(\vec{vu}))$. A *vazão* $\gamma_{\vec{\mathcal{R}}}(u)$ de um nó de origem u é então a razão entre o saldo de *fluxo* saindo de u em um período do escalonamento e o valor desse período, π, ou seja, $\gamma_{\vec{\mathcal{R}}}(u) = \sum_{v \in N(u)}(\phi_{\vec{\mathcal{R}}}(\vec{uv}) - \phi_{\vec{\mathcal{R}}}(\vec{vu}))/\pi$. A *vazão total*, $\gamma_{\vec{\mathcal{R}}}$, é a soma das vazões dos nós de origem. Portanto, $\gamma_{\vec{\mathcal{R}}} = \sum_{u \in V_O} \gamma_{\vec{\mathcal{R}}}(u)$. Mais uma vez podemos omitir o índice da notação quando o escalonamento estiver claro pelo contexto.

O funcionamento do modelo que acabamos de descrever é melhor visualizado por meio de um exemplo de execução de um escalonamento, conforme ilustrado na Figura 2. O

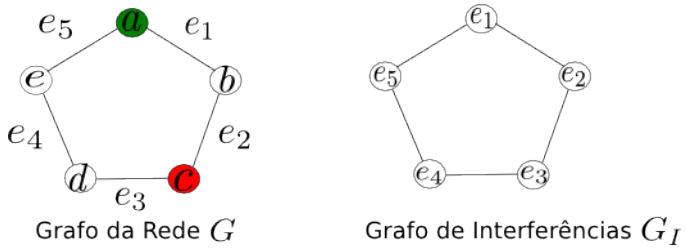

Figura 1 – Exemplo de rede modelada. O Grafo da Rede é um C_5 (ciclo com 5 vértices). O nó a é nó de origem. O nó c é o destino das mensagens.

Grafo da Rede e o Grafo de Interferências são ilustrados na Figura 1. As rodadas $R_1 = \{\vec{ae}, \vec{bc}\}$, $R_2 = \{\vec{ed}\}$ e $R_3 = \{\vec{ab}, \vec{dc}\}$ são repetidas continuamente, nesta ordem. Quando uma rodada é executada, cada nó v envia uma mensagem presente em sua memória de retenção de dados (se houver alguma) ou, caso $v \in V_O$, uma mensagem originada em v. É possível que, na execução de uma rodada, um nó $v \notin V_O$ não tenha mensagem para enviar em uma ativação de $\vec{vw}$, $w \in N(v)$. Porém, como a operação da rede consiste em execuções repetidas de um escalonamento, a partir de algum momento da operação, todas as transmissões $\vec{uv}, uv \in E$, podem ser executadas (sempre será possível enviar uma mensagem de u para v) quando requisitadas. Portanto, isso não influencia na vazão total. No exemplo da Figura 2, o tamanho do período do escalonamento é $\pi = 3$ e a vazão do nó de origem é $\gamma(a) = 2/3$, igual à vazão total γ. Observe que cada um dos nós b, d e e consome uma unidade de memória de retenção de dados. Isso se dá porque cada um deles precisa reter a mensagem recebida em uma rodada para envio em uma rodada posterior. Por exemplo, a mensagem que e recebe em R_1 é retida até ser enviada na rodada R_2.

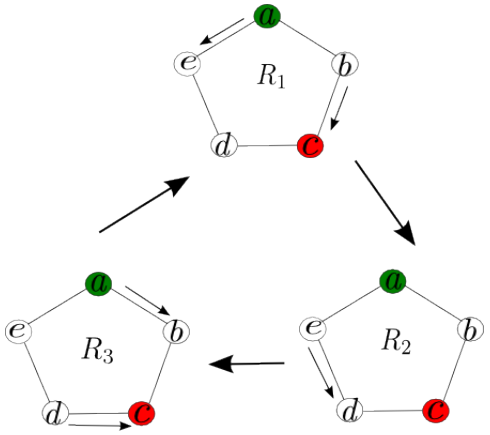

Figura 2 – Exemplo de execução de um escalonamento formado pelas rodadas $R_1 = \{\vec{ae}, \vec{bc}\}$, $R_2 = \{\vec{ed}\}$ e $R_3 = \{\vec{ab}, \vec{dc}\}$ para o exemplo da Figura 1. O sentido de cada transmissão é indicado por setas. O nó a é nó de origem. O nó c é o destino das mensagens.

Em um outro exemplo, ilustrado pela Figura 3, temos a execução de um escalonamento formado pelas rodadas $R_1 = \{\vec{ae}, \vec{bc}\}$, $R_2 = \{\vec{ed}, \vec{bc}\}$ e $R_3 = \{\vec{ab}, \vec{dc}\}$, nesta ordem, em uma

rede com dois nós de origem e um de destino. O tamanho do período do escalonamento é 3. Ao executarmos o escalonamento, obtemos uma vazão de $\gamma(a) = 2/3$ para o nó de origem a e $\gamma(b) = 1/3$ para o nó de origem b. A vazão total é então $\gamma = 2/3 + 1/3 = 1$. Observe que qualquer ordem de execução das rodadas produz um escalonamento com os mesmos valores de vazão (KLASING; MORALES; PéRENNES, 2008). Porém, a ordem afeta o consumo de memória de retenção de dados (fato tratado com mais detalhes na Seção 1.4). Para o escalonamento da Figura 3, a quantidade de memória necessária é de apenas uma unidade (cada nó armazena no máximo uma mensagem).

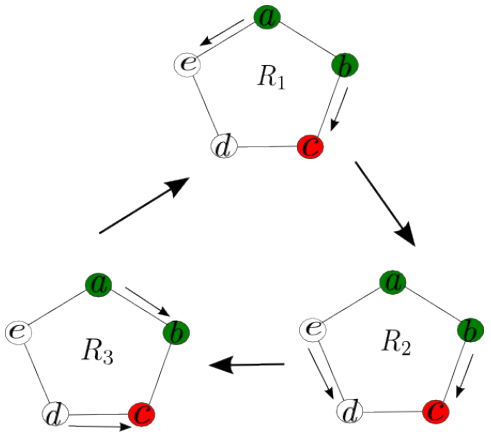

Figura 3 – Exemplo de execução de um escalonamento formado pelas rodadas $R_1 = \{\vec{ae}, \vec{bc}\}$, $R_2 = \{\vec{ed}, \vec{bc}\}$ e $R_3 = \{\vec{ab}, \vec{dc}\}$. O sentido de cada transmissão é indicado por setas. Os nós a e b são nós de origem. O nó c é o destino das mensagens das origens.

A operação da rede, segundo um escalonamento $\vec{\mathcal{R}} = \{R_1, \ldots, R_n\}$, estabelece caminhos entre pares origem-destino. Cada um desses caminhos, digamos P, contém um nó de origem v e um nó de destino w, além dos nós que definem transmissões ativadas em rodadas distintas. Se $\{v = v_1, v_2, \ldots, v_p = w\}$ é o conjunto dos vértices de P, então $P = \langle v_1\vec{v}_2, \ldots, v_{k-1}\vec{v}_p\rangle$ e a cada transmissão $v_i\vec{v_{i+1}}$, $i \in \{1, 2, \ldots, p-1\}$, associa-se uma rodada $R_{y(i)} \in \mathcal{R}$ que contém $v_i\vec{v_{i+1}}$ de tal forma que $i \neq i' \Rightarrow y(i) \neq y(i')$. Dessa forma, a execução das rodadas associadas com as transmissões em P gera uma sequência de ativações dessas transmissões. Na verdade, o número de sequência de ativações pode ser arbitrário pelo fato de a operação da rede consistir na execução repetida do escalonamento e, consequentemente, das rodadas associadas.

O Algoritmo 1 descreve como realizar o cálculo dos caminhos entre pares origem-destino a partir de um escalonamento. O conjunto de caminhos em construção é denotado por $\mathcal{F}$, enquanto que o conjunto de caminhos já construídos é denotado por $\mathcal{P}$. A variável k indica o número de rodadas percorridas, o que significa o tempo decorrido na execução do escalonamento $\vec{\mathcal{R}}$. Para cada caminho P, l_P indica o momento em que o último arco de P foi adicionado a ele. A ideia do algoritmo é percorrer as rodadas do escalonamento na ordem estabelecida, iterativamente, e construir os caminhos origem-destino. Quando

Algoritmo 1: Algoritmo para cálculo dos caminhos origem-destino de um escalonamento.

Chamada: CAMINHOS($\vec{\mathcal{R}} = \{R_1, \ldots, R_n\}$,$V_O$,$V_D$)
Entrada: Escalonamento $\vec{\mathcal{R}}$ e conjuntos de nós de origem e de destino, V_O e V_D.
Resultado: Caminhos origem-destino na rede.

```
1  F ← ∅, P ← ∅, k ← 0
2  repita
3      para i = 1, ..., n faça
4          para todo uv⃗ ∈ R_i faça
5              F_u ← {P ∈ F | P = P'u}
6              se F_u ≠ ∅ ou u ∈ V_O então
7                  se F_u ≠ ∅ então
8                      Seja P ∈ F_u tal que l_P é mínimo
9                      P ← Puv⃗
10                 senão
11                     Criar novo caminho P = uv⃗ e incluir em F
12                 l_P ← k
13                 se v ∈ V_D então
14                     F ← F\{P}
15                     P ← P ∪ P
16                 R_i ← R_i\{uv⃗}
17         k ← k + 1
18 até  R_i = ∅, i = 1, ..., n
19 retorne P
```

uma rodada é percorrida, todas as suas transmissões são analisadas para determinar em qual caminho elas serão incluídas.

Para determinar o caminho que inclui uma transmissão $\vec{uv}$ de uma rodada R_i, primeiro verifica-se a existência de algum caminho em construção em que o último vértice é u (o conjunto desses caminhos é denotado por $\mathcal{F}_u$). Se existe pelo menos um caminho satisfazendo essas condições, então o caminho escolhido P é o caminho em que seu último arco foi adicionado antes dos outros caminhos em $\mathcal{F}_u$. Isso corresponde ao envio em $\vec{uv}$ ser referente à mensagem que está a mais tempo armazenada na memória de retenção de dados de u (satisfazendo o tipo de fila de saída em *FIFO*). Então, o arco $\vec{uv}$ é adicionado a P, a l_P é atribuído o valor do tempo atual (número de rodadas já percorridas até então) e a transmissão $\vec{uv}$ é retirada da rodada R_i. Se não existe tal caminho, é verificado se u é nó de origem, o que corresponde ao envio de uma nova mensagem através de $\vec{uv}$, então um novo caminho P é criado partindo de u, sua variável de tempo l_P é inicializada com o tempo atual e a transmissão $\vec{uv}$ é retirada da rodada R_i. Após a adição do arco $\vec{uv}$ em um caminho, é verificado se v é um nó de destino, o que determina o fim da construção do caminho e ele é então adicionado ao conjunto $\mathcal{P}$. Depois da análise da última rodada

do escalonamento, o processo é novamente iniciado a partir da primeira rodada R_1. Isso é feito até que todas as rodadas se tornem vazias.

Para exemplificar o funcionamento do Algoritmo 1, considere o exemplo de escalonamento ilustrado pela Figura 2. Primeiramente, os conjuntos de caminhos construídos e em construção são vazios. Ao analisarmos a rodada R_1 verificamos que podemos criar um novo caminho iniciado com o arco $\vec{ae}$ e nada é feito em relação à transmissão $\vec{bc}$, pois o conjunto de caminhos construídos é vazio. Dessa forma, a rodada R_1 agora contém apenas a transmissão $\vec{bc}$. Para a rodada R_2, verificamos que existe um caminho em construção que termina no vértice e e então adicionamos o arco $\vec{ed}$ a esse caminho. Na rodada R_3, podemos adicionar o arco $\vec{dc}$ ao caminho $\{\vec{ae}, \vec{ed}\}$ e como c é nó de destino, o caminho formado é então $\{\vec{ae}, \vec{ed}, \vec{dc}\}$. Além disso, um novo caminho, iniciado pelo arco $\vec{ab}$, é criado, pois o conjunto de caminhos em construção que terminam com o vértice a é vazio. Neste momento, apenas a rodada R_1 não é vazia e ela contém apenas a transmissão $\vec{bc}$ que pode ser adicionada ao caminho em construção $\{\vec{ab}\}$ formando o segundo caminho $\{\vec{ab}, \vec{bc}\}$, pois c é nó de destino. Observe que em cada um dos caminhos, a quantidade de informação escoada equivale a uma unidade, ou uma mensagem.

Os dois problemas mencionados inicialmente neste capítulo podem ser vistos como problemas de otimização formulados em um modelo de comunicações em uma rede de rádio conforme apresentado anteriormente. Eles incluem restrições e critérios de otimização. Naturalmente, o critério de otimização corresponde a alguma forma de uso eficiente dos recursos da rede.

O primeiro problema, chamado problema de *Ponderação de Rodadas* (*PR*) (GOMES, 2009) (KLASING; MORALES; PéRENNES, 2008), diz respeito a comunicações ponderadas por demandas relativas nos nós de origem e o segundo problema, chamado problema de *Escalonamento de Caminhos* (*EC*) (VIEIRA et al., 2012), diz respeito a comunicações restritas a caminhos pré-definidos. Ambos os problemas têm como critério de otimização a maximização da vazão total através de um escalonamento a ser executado repetidamente (operação da rede). Porém, há restrições que limitam a vazão de cada nó em cada um dos problemas. Nas seções seguintes, apresentamos uma descrição formal para cada um dos problemas.

1.3 Problema de Ponderação de Rodadas

Uma forma de operação da rede que ocorre com frequência caracteriza-se pela geração de informações a taxas distintas entre os nós de origem. Para modelar tal situação, atribui-se a cada nó de origem $u \in V_O$ uma demanda $d(u)$ de mensagens a enviar através da rede e supõe-se que o tamanho da memória de retenção de dados seja ilimitado para todos os nós. Este é o problema com relação ao qual esta dissertação traz contribuições. Portanto, além desta introdução, ele é mencionado nos demais capítulos deste texto.

O problema PR recebe como entrada os seguintes elementos:

1. Grafo da Rede $G = (V, E)$;
2. Conjuntos $V_O, V_D \subset V$, de nós de origem e destino, respectivamente;
3. Grafo de Interferências $G_I = (E, I)$;
4. Função de demanda $d : V_O \to \mathbb{N}\backslash\{0\}$, que indica a demanda de comunicação de cada nó de origem.

Uma solução para o problema é um protocolo $\mathcal{R}$ tal que o fluxo gerado por cada nó de origem u satisfaz a sua demanda $d(u)$ um número inteiro de vezes. Mais especificamente, o protocolo $\mathcal{R}$ deve satisfazer $\sum_{v \in N(u)} \phi_{\mathcal{R}}(\vec{uv}) - \phi_{\mathcal{R}}(\vec{vu}) \geq k \cdot d(u)$, para algum $k \in \mathbb{N}$ e para todo $u \in V_O$. O "termo ponderação de rodadas"é usado para indicar a quantidade de vezes que a mesma aparece no protocolo $\mathcal{R}$. O critério de otimização do problema PR é a maximização da vazão total, o que corresponde a maximizar a relação k/π. Assim sendo, o problema PR consiste em determinar um protocolo tal que a execução continuada de um escalonamento qualquer atenda a demanda pelo menor tempo médio possível.

Considere novamente o exemplo da Figura 1 desta vez visto como uma instância do problema PR. Seja $d(a) = 2$ a demanda do nó de origem a. Uma solução para esse exemplo é ilustrada na Figura 4. O fluxo de cada aresta de G é de uma unidade. As rodadas R_1, R_2 e R_3 (que satisfazem a demanda do vértice de origem uma única vez) são repetidas continuamente gerando uma vazão de $2/3$ e tempo médio de atendimento da demanda de 3 unidades.

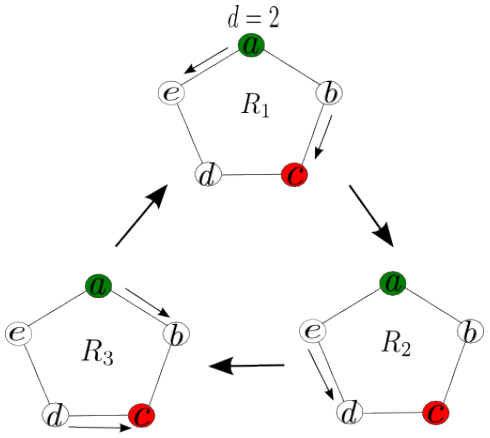

Figura 4 – Exemplo de execução de um escalonamento formado pelas rodadas R_1, R_2 e R_3 em um grafo de rede C_5 com arestas numeradas de 1 a 5. O sentido de cada transmissão é indicado por setas. A demanda é de 2 unidades no nó a. O nó c é o destino das mensagens.

Uma solução com valor maior é mostrada na Figura 5. O fluxo de cada aresta de G é de duas unidades. As rodadas R_1, R_2, R_3, R_4 e R_5 (que satisfazem a demanda do vértice

de origem duas vezes) são repetidas continuamente gerando uma vazão de 4/5 e tempo médio de atendimento da demanda igual a $5/2 = 2.5$.

A particularidade do problema *PR* está no fato de envolver uma combinação de um problema de coloração fracionária (em G_I) com um problema de fluxo (em G). Nessa combinação, a coloração está associada às restrições de interferência, enquanto o fluxo está associado à restrição de atendimento das demandas dos nós de origem. Naturalmente, a coloração é a "parte difícil"para os algoritmos de resolução pois trata-se de um problema NP-Difícil (KLASING; MORALES; PéRENNES, 2008). Esta questão é tratada em detalhes no Capítulo 4, no qual a metodologia adotada é a realização de um estudo sobre métodos de resolução desse problema. Por haver uma relação entre o problema *PR* e colorações fracionárias de G_I, estudamos também o problema de *Coloração Fracionária* (*CF*) em detalhes nos Capítulos 2 e 3, sobretudo as suas estratégias de resolução através de heurísticas e de programação inteira.

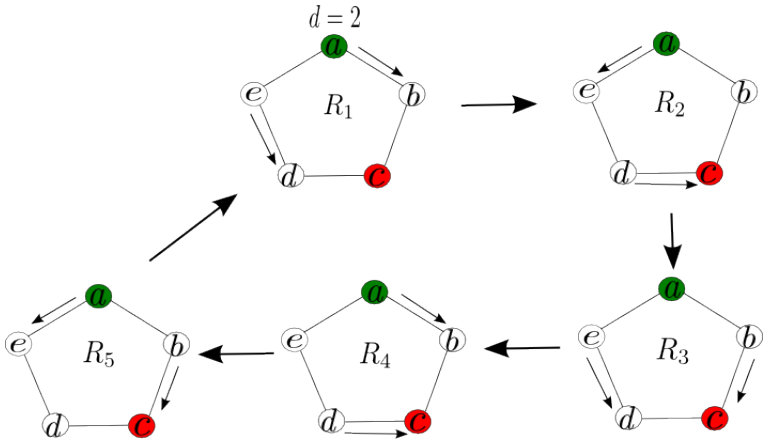

Figura 5 – Exemplo de uma solução melhor para o exemplo da Figura 1 com demanda igual a 2 para o nó *a*. O escalonamento é formado por $\langle R_1, R_2, R_3, R_4, R_5 \rangle$.

1.4 Problema de Escalonamento de Rodadas

Em muitas aplicações, busca-se otimizar um ou mais aspectos que caracterizam o consumo de recursos durante uma operação da rede. Um dentre os vários aspectos da operação de uma *RRM* que não são levados em consideração no problema *PR* é a quantidade de memória de retenção de dados. Nesse sentido, um problema complementar ao problema *PR* é o problema de *Escalonamento de Rodadas* (*ER*), que recebe como entrada um protocolo $\mathcal{R}$ e tem como soluções os possíveis escalonamentos de $\mathcal{R}$. O critério de otimização é a minimização do total de memória de retenção de dados utilizada por qualquer nó da rede. Uma estratégia de otimização de operação da *RRM* consiste em resolver sucessivamente os problemas *PR* e *ER*, estratégia essa que parece adequada quando o aspecto mais importante na comunicação na rede é a vazão de dados. O problema *ER*

não é abordado nesta dissertação. Sua menção neste capítulo de introdução ocorre com o intuito de ampliar o contexto do problema *PR*. O leitor interessado no problema *ER* pode consultar (FARIAS, 2014).

Dado um protocolo $\mathcal{R}$ viável para uma instância do problema *PR* e uma ordenação das rodadas de $\mathcal{R}$ formando um escalonamento $\vec{\mathcal{R}}$, denotamos por $m(v)$ o número máximo de mensagens retidas simultaneamente por um nó v, $v \notin V_D$, quando $\vec{\mathcal{R}}$ é executado repetidamente e $M = \max_{v \in V - V_D} m(v)$ o total de memória utilizada na rede.

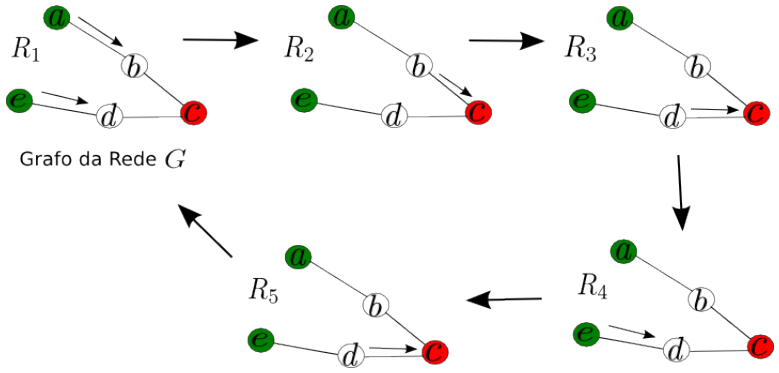

Figura 6 – Exemplo de execução de um escalonamento formado pelas rodadas $R_1 = \{\vec{ab}, \vec{ed}\}$, $R_2 = \{\vec{bc}\}$, $R_3 = \{\vec{dc}\}$, $R_4 = \{\vec{ed}\}$ e $R_5 = \{\vec{dc}\}$ em um Grafo da Rede com vértices nomeados de a a e. O sentido de cada transmissão é indicado por setas. Os nós a e e são nós de origem. O nó c é o destino das mensagens das origens.

Considere o exemplo ilustrado pela Figura 6. Temos a execução de um escalonamento $\langle R_1, R_2, R_3, R_4, R_5 \rangle$ em uma rede com dois nós de origem e um de destino. O período do escalonamento é 5. Ao executarmos o escalonamento, obtemos uma vazão de $1/5$ para o nó de origem a e $2/5$ para o nó de origem e. A vazão total é então $1/5 + 2/5 = 3/5$. Como já mencionado, qualquer ordem de execução das rodadas produz um escalonamento com os mesmos valores de vazão. Porém, o consumo de memória de retenção de dados é afetado. Para o escalonamento da Figura 6, a quantidade de memória de retenção de dados necessária é de apenas uma unidade para cada nó (cada nó armazena no máximo uma mensagem), o que nos dá $M = 1$. Se executássemos as rodadas na ordem $\langle R_1, R_4, R_2, R_3, R_5 \rangle$, seria necessária uma memória de retenção de dados com capacidade de armazenamento de duas mensagens para o nó d e uma mensagem para os demais nós, resultando em $M = 2$.

1.5 Problema de Escalonamento de Caminhos

Um problema não tratado diretamente nesta dissertação, mas usado para avaliação dos algoritmos propostos para o problema *PR* no Capítulo 5, é o problema de *Escalonamento de Caminhos* (*EC*). A entrada do problema *EC* inclui uma coleção de caminhos que são

usados para estabelecer restrições sobre os protocolos devido à capacidade limitada de memória nos nós da rede. Esses caminhos têm uma forma particular. Sejam $P_1, P_2, \ldots, P_p$ (não necessariamente disjuntos ou diferentes) caminhos origem-destino simples (caminhos que não visitam um vértice mais de uma vez) em G. O conjunto de vértices de cada caminho P_i é denotado por X_i, $i \in \{1, \ldots, p\}$, enquanto o conjunto de transmissões é denotado por Y_i. Para cada $i, j \in \{1, 2, \ldots, p\}$, um elemento de Y_i e um elemento de Y_j são sempre distinguíveis (se referem a transmissões diferentes), mesmo quando correspondem a arestas com os mesmos extremos. Sejam $X = \bigcup_{i=1}^{p} X_i$ e $Y = \bigcup_{i=1}^{p} Y_i$. Um exemplo de instância para o problema *EC* é ilustrado na Figura 7.

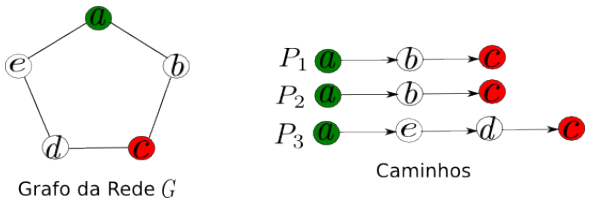

Figura 7 – Exemplo de instância do problema *EC*. O Grafo da Rede é um C_5 com arestas numeradas de 1 a 5 e vértices nomeados de a a e. A coleção de caminhos é formada por $P_1 = \{\vec{ab}, \vec{bc}\}$, $P_2 = \{\vec{ab}, \vec{bc}\}$ e $P_3 = \{\vec{ae}, \vec{ed}, \vec{dc}\}$.

Os escalonamentos viáveis no problema *EC* dependem da forma em que os caminhos $P_1, P_2, \ldots, P_p$ são usados. Uma solução para o problema é um escalonamento em que cada elemento de Y esteja incluído em pelo menos uma das rodadas do protocolo correspondente e que cada nó não retenha, simultaneamente, mais do que uma mensagem em trânsito passando através do mesmo para cada caminho origem-destino que passe por ele. Por isso, a ordem de execução das rodadas definida pelo escalonamento é importante.

O problema *EC* recebe como entrada os seguintes elementos:

1. O Grafo da Rede $G = (V, E)$;
2. Uma coleção de caminhos origem-destino $P_1, P_2, \ldots, P_p$;
3. O Grafo de Interferências $G_I = (E, I)$;

O critério de otimização do problema *EC* é a maximização da vazão total. Considere o exemplo ilustrado pela Figura 7 como instância do problema *EC*. A coleção de caminhos é formada por $P_1 = \{\vec{ab}, \vec{bc}\}$, $P_2 = \{\vec{ab}, \vec{bc}\}$ e $P_3 = \{\vec{ae}, \vec{ed}, \vec{dc}\}$. Um exemplo de solução para essa instância é o escalonamento ilustrado pela Figura 5. O fluxo no período tem valor 1 nos caminhos P_1 e P_2 e valor 2 no caminho P_3. Maiores detalhes sobre o problema *EC* podem ser encontrados em (VIEIRA et al., 2012).

1.6 Abordagens Existentes para Resolução do Problema PR

A questão central desta dissertação é a obtenção de boas soluções para o problema *PR*. Uma revisão bibliográfica cuidadosa revela que as abordagens encontradas na literatura não são completamente satisfatórias. Essa revisão bibliográfica envolve não somente o problema *PR*, mas também trabalhos sobre coloração fracionária, na perspectiva da possibilidade de adaptação dos algoritmos para o problema *CF* para o problema *PR*. A seguir, detalhamos a análise que realizamos dos trabalhos na literatura.

O método mais explorado para solução do problema *CF* consiste em formular o problema como relaxação linear de uma formulação de programação inteira para o problema de coloração inteira (essa formulação é apresentada no Capítulo 2). Por se tratar de uma relaxação linear, o uso de soluções de coloração inteira para aproximação de colorações fracionárias ótimas é uma alternativa ruim. Por isso, estuda-se técnicas de resolução da formulação para do problema *CF*. A maior dificuldade nesse método reside no fato de a formulação possuir um número exponencial de variáveis. Para contornar parcialmente essa dificuldade, um método de geração de colunas foi inicialmente proposto por (MEHROTRA; TRICK, 1996) e possui resultados significantes. Posteriores aperfeiçoamentos desse método, que lidam com introdução e propagação de cortes, são propostos em (GUALANDI; MALUCELLI, 2012), (HANSEN; LABBÉ; SCHINDL, 2009), (MALAGUTI; MONACI; TOTH, 2011). Porém, a implementação desse método usando pacotes de otimização em ponto flutuante para problemas de programação linear inteira, como o `CPLEX`, pode produzir erros numéricos. Em (HELD; COOK; SEWELL, 2012), é mostrado como implementar o método de forma a evitar tais erros. Entretanto, a perda de eficiência em tempo de execução em relação ao uso de solvers de ponto flutuante é significativa.

Em (GOMES, 2009), o método de geração de colunas é adaptado para o problema *PR*. Embora bastante estudado, e com resultados satisfatórios para o cálculo do número cromático fracionário (ou da vazão no caso do problema *PR*), a abordagem por geração de colunas não fornece diretamente uma solução. Para visualizar esse fato, voltemos ao exemplo da Figura 5. Conforme já observado, a vazão obtida na solução indicada vale 4/5 (e é ótima). O método de geração de colunas sobre a formulação apresentada no Capítulo 2 obtém essa mesma solução, porém expressa como a soma de valores fracionários. Portanto, os valores inteiros de ocorrências das transmissões e o valor do período não estão disponíveis diretamente. Uma forma de obter esses valores de que necessitamos seria usar um programa de resolução de programação linear exata (por meio de manipulação de frações como valores das variáveis). No entanto, o programa conhecido com tal propriedade possui tempo de cálculo muito alto (HELD; COOK; SEWELL, 2012).

A natureza do problema de *Ponderação de Rodadas* demanda não somente o valor da solução ótima, mas também a solução ótima em si, especificando completamente todas as rodadas utilizadas. Por isso, desejamos não apenas obter o *número cromático fracionário*

ponderado de um grafo, mas também a coloração fracionária ponderada que resulta no valor ótimo. Tal valor ótimo para o problema *PR* é uma versão ponderada do *número cromático fracionário* e o problema de determiná-lo é considerado NP-Difícil (KLASING; MORALES; PéRENNES, 2008).

Uma alternativa ao método exato de geração de colunas é o uso de heurísticas. Uma heurística para coloração fracionária é proposta por Balas e Xue em (BALAS; XUE, 1996). Essa é a única heurística conhecida por nós que fornece uma solução para coloração fracionária, além da que desenvolvemos nesta pesquisa. Trata-se de uma heurística gulosa proposta no contexto de algoritmos para resolução do problema de clique máxima. Conforme indicado em (BALAS; XUE, 1996), o problema *CF* é o dual da relaxação linear do problema de clique máxima. Dessa forma, os autores propõem usar colorações fracionárias para estabelecer limites superiores em um algoritmo *branch-and-bound*. Nesse contexto, um dos objetivos dessa heurística é que o seu tempo de execução seja relativamente pequeno, embora a solução encontrada, em muitos casos, não seja de boa qualidade.

Na literatura, há um algoritmo de planos-de-corte, usado para tratar a questão do número exponencial de restrições na formulação por representantes, proposto por (CAMPêLO; CAMPOS; CORRêA, 2009) para determinar limites inferiores para o problema *CF*. Além disso, são mostradas heurísticas de separação para desigualdades válidas associadas a cliques maximais e buracos ímpares de subgrafos do grafo de instância do problema.

1.7 Resultados Obtidos

Como resultado desta pesquisa, implementamos uma heurística baseada em relaxação lagrangeana que desenvolvemos para o problema *CF*. Essa heurística é inspirada na heurística *FCP* (*Fractional Coloring Procedure*) (BALAS; XUE, 1996). Uma formulação por representantes, conforme proposta em (CAMPêLO; CAMPOS; CORRêA, 2009), é usada para a determinação de limites inferiores, que são aplicados na atualização dos multiplicadores lagrangeanos nos métodos de subgradiente para encontrar um bom limite dual. Tal formulação envolve um número polinomial de variáveis em detrimento do número exponencial de restrições e, além disso, nos permite decompor o problema relaxado (relaxação lagrangeana) para fazer uso de paralelismo com o intuito de obter uma resolução mais rápida. Os resultados da nossa implementação são comparados com os da heurística *FCP* e apresentados no Capítulo 5.

Para o problema *PR*, a mesma técnica de resolução adotada para o problema *CF* foi realizada. Neste caso, a heurística lagrangeana é baseada na heurística *WFCP* (*Weighted Fractional Coloring Procedure*) (BALAS; XUE, 1996), que é usada para o caso ponderado do problema *CF* em que cada vértice pode receber quantidade de cores diferentes em uma solução ótima (ver Capítulo 2). Para o problema *PR*, a relaxação lagrangeana do problema *CF* é acrescida de um subproblema de fluxo que é facilmente resolvido por

um algoritmo combinatório polinomial. Uma comparação de resultados é feita com um algoritmo que resolve heuristicamente o problema *EC* no Capítulo 5.

1.8 Organização do Texto

O restante do texto está dividido como segue. No Capítulo 2, apresentamos formulações por conjuntos independentes para os problemas de *Coloração Inteira* e *Coloração Fracionária*, incluindo também heurísticas gulosas para os problemas de coloração. Notação de modelagem por representantes e técnicas de resolução de problemas de otimização usadas ao longo do texto são apresentadas no Capítulo 3. Além disso, uma relaxação lagrangeana para o problema *CF* e uma heurística baseada nessa relaxação são também mostradas. No Capítulo 4, apresentamos uma formulação por representantes para o problema *PR* juntamente a uma técnica de resolução baseada em uma heurística lagrangeana que também desenvolvemos, adaptada da técnica utilizada para o problema *CF*. Finalmente, mostramos os resultados computacionais das nossas implementações e análises dos mesmos para os problemas *CF* e *PR* no Capítulo 5 e conclusões no Capítulo 6.

2 Formulações de Programação Linear Inteira e Heurísticas Gulosas para Problemas de Coloração

Este capítulo é dedicado a traçar um panorama das principais técnicas algorítmicas para lidar com problemas de coloração. Duas abordagens são destacadas. A primeira baseia-se em uma formulação de programação linear inteira. Neste caso, detalhamos uma formulação e fazemos uma rápida descrição das principais ideias exploradas nos algoritmos exatos baseados nessa formulação. Em particular, mostramos em detalhes a dificuldade de transformar uma solução com valores fracionários em uma coloração. A apresentação de uma formulação alternativa é deixada para o Capítulo 3. A segunda abordagem é constituída de heurísticas gulosas para obtenção de soluções aproximadas. A apresentação privilegia aquelas heurísticas que usamos em capítulos posteriores para construção de colorações. O conteúdo deste capítulo envolve tanto coloração inteira quanto coloração fracionária de grafos arbitrários. Neste capítulo, denotamos um grafo arbitrário por $G = (V, E)$, e as instâncias de grafos mencionadas pertencem ao conjunto de instâncias de coloração de grafos da DIMACS, encontradas em http://mat.gsia.cmu.edu/COLOR/instances.html.

2.1 Definições

Uma *coloração inteira* de G é uma atribuição de cores aos vértices de G tal que cada vértice recebe pelo menos uma cor e as cores atribuídas a vértices adjacentes são todas distintas. Uma *coloração k-dimensional*, $k \in \mathbb{N}$, em G é uma atribuição de cores aos vértices de G tal que cada vértice recebe pelo menos k cores distintas e vértices adjacentes recebem conjuntos de cores disjuntos. O valor $\chi_k(G)$ representa o menor número de cores distintas utilizadas em uma coloração k-dimensional. Quando $k = 1$, $\chi(G) = \chi_1(G)$ é conhecido como *número cromático* de G, e o problema de determiná-lo é conhecido como problema de *Coloração Inteira* (*CI*).

O problema de *Coloração Fracionária* (*CF*) em G consiste em determinar um valor k e uma coloração k-dimensional em G que minimizem a razão $\chi_k(G)/k$. O valor ótimo $\chi_F(G)$ para esse problema é chamado de *número cromático fracionário*. Sabe-se que determinar $\chi_F(G)$ é um problema NP-difícil (LUND; YANNAKAKIS, 1994).

Para a versão ponderada dos problemas definidos anteriormente, cada vértice $u \in V$ recebe uma ponderação $p(u)$ como entrada que indica a quantidade relativa de cores que u deve receber. Uma *coloração k-dimensional ponderada*, $k \in \mathbb{N}$, em G é uma atribuição

de cores aos vértices de G tal que cada vértice u recebe pelo menos $k \cdot p(u)$ cores distintas e vértices adjacentes recebem conjuntos de cores disjuntos. O valor ${\chi^P}_k(G)$ representa o menor número de cores distintas utilizadas em uma coloração k-dimensional ponderada. A versão ponderada do problema de *Coloração Fracionária* é chamada de problema de *Coloração Fracionária Ponderada* (*CFP*) e consiste em determinar um valor k e uma coloração k-dimensional ponderada em G que minimizem a razão ${\chi^P}_k(G)/k$. O valor ótimo ${\chi^P}_F(G)$ para esse problema é chamado de *número cromático fracionário ponderado.*

Ao longo do texto, denotamos por $N(u) = \{v \in V \mid uv \in E\}$ a vizinhança de um vértice $u \in V$ em $G = (V, E)$.

2.2 Modelagem por Conjuntos Independentes

Para construir um modelo de programação linear inteira para o problema *CF*, definimos $\mathcal{S}$ como sendo o conjunto de todos os conjuntos independentes maximais de G e $\omega : \mathcal{S} \to \{0, 1\}$ uma *ponderação* binária desses conjuntos independentes. Essa ponderação indica, para cada conjunto independente $S \in \mathcal{S}$, se S é escolhido em uma coloração de G, dada por ω, onde cada conjunto independente maximal representa uma cor. Denotamos por $\mathcal{S}(u) = \{S \in \mathcal{S} \mid u \in S\}$, para $u \in V$, o conjunto dos conjuntos independentes que incluem o vértice u. Dessa forma, uma formulação para o problema *CI* (MEHROTRA; TRICK, 1996) é

$$
\begin{aligned}
(IS) \quad \min \quad & \textstyle\sum_{S \in \mathcal{S}} \omega(S) && && (2.1)\\
\text{s.a.} \quad & \textstyle\sum_{S \in \mathcal{S}(u)} \omega(S) \geq 1 && u \in V && (2.2)\\
& \omega(S) \in \{0, 1\} && S \in \mathcal{S}
\end{aligned}
$$

As restrições (2.2) garantem que cada vértice recebe pelo menos uma cor e a função objetivo (2.1) minimiza o número de cores usadas na coloração dada por ω para determinar $\chi(G) = \sum_{S \in \mathcal{S}} \omega(S)$. Observe que uma solução viável dessa formulação é uma coloração 1-dimensional de G em que alguns vértices podem receber mais de uma cor. Uma coloração inteira com exatamente uma cor por vértice pode ser obtida removendo cada vértice de todas as cores que o incluem exceto uma delas. A relaxação linear da formulação (IS) é uma formulação para o problema *CF*. Para verificar esse fato, começamos escrevendo a seguinte formulação diretamente da definição do problema *CF*:

$$\begin{array}{lll} \min & \sum_{S\in\mathcal{S}} \frac{1}{k}\omega(S) & \\ \text{s.a.} & \sum_{S\in\mathcal{S}(u)} \omega(S) \geq k & u \in V \\ & \omega(S) \leq k & S \in \mathcal{S} \\ & \omega(S) \in \mathbb{N} & S \in \mathcal{S} \\ & k \in \mathbb{N} & \end{array} \tag{2.3}$$

Nesse caso, usamos uma variável inteira $\omega(S) \in \mathbb{N}$ para cada conjunto independente $S \in \mathcal{S}$ de G, que representa o número de vezes que o conjunto independente é escolhido na solução ω. As restrições (2.3) impõem que cada vértice $u \in V$ seja coberto por pelo menos k conjuntos independentes, em que k é uma variável inteira representando o número de cores que cada vértice deve receber. Assim sendo, se ω satisfaz essas restrições, e removemos cada vértice de todas as cores que o incluem exceto k delas, obtemos uma coloração k-dimensional. Minimizando a razão do número de cores distintas sobre número de cores que cada vértice recebe, dada por $\sum_{S\in\mathcal{S}} \frac{1}{k}\omega(S)$, temos então uma coloração fracionária ótima.

Utilizando a transformação de variáveis em que $\omega(S)$ passa a significar $\frac{\omega(S)}{k}$, conseguimos descrever a versão linear da formulação mostrada anteriormente:

$$(FIS) \quad \min \quad \sum_{S\in\mathcal{S}} \omega(S) \tag{2.4}$$

$$\text{s.a.} \quad \sum_{S\in\mathcal{S}(u)} \omega(S) \geq 1 \qquad u \in V \tag{2.5}$$

$$\omega(S) \leq 1 \qquad S \in \mathcal{S}$$

$$\omega(S) \in \mathbb{Q}_+ \qquad S \in \mathcal{S} \tag{2.6}$$

As restrições de racionalidade (2.6) podem ser eliminadas visto que todo vértice de um poliedro racional (poliedro que pode ser descrito por matriz e termos independentes racionais) é racional. A prova disso é derivada da programação linear, onde temos que os vértices de um poliedro são equivalentes a soluções básicas viáveis, que são determinadas por uma expressão envolvendo operações básicas entre elementos racionais (BREGALDA; BORNSTEIN; OLIVEIRA, 1981).

O fato de o problema (FIS) corresponder à relaxação linear do problema (IS) estabelece uma ligação entre os dois problemas explorada em vários algoritmos. Um exemplo é a estratégia de resolução exata do problema (IS) usando *branch-and-bound* proposta em (MEHROTRA; TRICK, 1996). Nessa estratégia, o número cromático fracionário é usado como limite inferior no percurso da árvore de busca. Sendo assim, em cada nó é resolvido um problema (FIS) para um certo subgrafo definido pelas ramificações. A questão do número de variáveis potencialmente exponencial é tratada com o uso do método de geração de colunas, onde os subproblemas escravos correspondem a problemas de conjunto independente máximo ponderado.

O fato relevante nesse trabalho é que a construção da solução do problema (FIS) é deixado em aberto. Além da dificuldade de encontrar uma solução com valor fracionário a cada variável das formulações propostas para o problema, há a dificuldade de transformá-la em uma coloração fracionária, definindo os conjuntos de cores usados na solução ótima. Por isso, estudamos o uso de heurísticas para o problema *CF*. Pela definição dos problemas *CF* e *CI*, podemos observar um possível gap muito grande entre o número cromático e o número cromático fracionário para alguns grafos. Como exemplo, o grafo `myciel4`, de número cromático igual a 5, possui número cromático fracionário igual a 3.24.

2.3 Heurística Gulosa para Coloração Inteira

Uma heurística gulosa muito conhecida para coloração inteira de um grafo é a heurística *DSATUR* (*Degree SATURation*) (BRÉLAZ, 1979). Essa heurística pode ser utilizada para gerar a coloração inteira em cada iteração da heurística de coloração fracionária da Seção 2.4. Na descrição de algoritmos de coloração, usamos os termos "cor", "classe de cor"e "conjunto independente"indistintamente.

Uma heurística gulosa constrói uma coloração de G iterativamente. Inicialmente, uma ordem entre os vértices é estabelecida. Em seguida, os vértices são percorridos nessa ordem, atribuindo a cada vértice v a menor cor não interditada pelas cores atribuídas aos vizinhos de v. Na heurística *DSATUR*, a ordem dos vértices é ajustada dinamicamente. Inicialmente, definem-se o número de cores disponíveis como sendo $\ell = 1$ e o conjunto independente correspondente $S_1 = \emptyset$. A cada iteração $k = 1, \ldots, n$, procede-se da seguinte maneira:

- Cada vértice recebe um *grau de saturação*, que é o número de cores distintas atribuídas a vértices em sua vizinhança em iterações precedentes. A atribuição de uma cor a um vértice, digamos v, ao final da iteração pode provocar uma alteração nos graus de saturação dos vizinhos de v.

- Os vértices não coloridos são ordenados pelo maior valor de grau de saturação corrente e, caso haja empate, o de maior grau (número de vértices na vizinhança) tem prioridade.

- Escolhe-se o primeiro vértice v na ordem estabelecida como indicado acima e atribui-se a menor cor possível S_k dentre $S_1, \ldots, S_\ell$ para v. Caso a cor escolhida seja $k = \ell$, incrementa-se ℓ de 1. Para todo $w \in N(v)$ tal que $N(w) \cap S_k = \{v\}$, o grau de saturação aumenta de uma unidade.

A heurística é gulosa devido ao critério de ordem em que os vértices são examinados, fazendo a escolha que parece ser a melhor no momento, em cada iteração. A técnica de atribuir a menor cor possível ao vértice escolhido em uma iteração é muito usada em

algoritmos de coloração. A complexidade do *DSATUR* é determinada pela complexidade de manutenção da ordem dos vértices e determinação da menor cor possível para cada vértice a ser colorido. O cálculo dos graus de todos os vértices de G se faz em $O(|V|^2)$. A ordenação inicial segundo esses vértices consome tempo em $O(|V| \cdot \log |V|)$. A cada iteração do algoritmo, a determinação da cor do vértice a ser colorido pode ser realizada em $O(|V|^2)$, pois para cada cor já criada é preciso verificar se existe algum vizinho com essa cor. Em seguida, a atualização dos graus de saturação e da ordenação dos vértices podem ser realizadas em $O(|V| \cdot \log |V|)$. Como a cada iteração exatamente um vértice é colorido, o número de iterações é $|V|$. Portanto, a complexidade do *DSATUR* é $O(|V|^3 + |V|^2 \cdot \log |V|)$ ou $O(|V|^3)$.

2.4 Heurística Gulosa para Coloração Fracionária

O Algoritmo 2 é conhecido como heurística *FCP* (*Fractional Coloring Procedure*) cuja ideia é calcular uma coloração k-dimensional, para alguns valores de k, e tomar, dentre essas colorações, a que possui o menor valor da razão (número total de cores/k) (BALAS; XUE, 1996). Para $k = 1$, suponha que tenhamos uma coloração 1-dimensional (coloração inteira) $\mathcal{C}_1 = \{C_1, ..., C_q\}$ de G, onde cada C_i é uma classe de cor. A coloração $\mathcal{C}_1$ é uma partição dos vértices de G em que algumas dessas classes de cor podem não ser maximais. Em seguida, passamos a $k = 2$ e tentamos estender as classes de cor de $\mathcal{C}_1$ nas linhas 3 – 7 de tal forma a atribuir mais uma cor, dentre as cores 1 até q, para cada vértice do grafo. Nessa etapa, seja U o conjunto dos vértices que não receberam a segunda cor. Se colorirmos o grafo $G[U]$ usando uma heurística de coloração inteira gerando uma coloração $\mathcal{C}_2$ (o que acontece na linha 8 do Algoritmo 2), encontramos então uma segunda cor para os vértices em U e, assim, todos os vértices de G recebem duas cores. Se $|\mathcal{C}_1| + |\mathcal{C}_2|/2 < |\mathcal{C}_1|$, então o valor da coloração fracionária $\mathcal{C}_1 \cup \mathcal{C}_2$ é menor do que o da coloração fracionária $\mathcal{C}_1$. A ideia é aplicar esse processo a cada iteração, incrementando em uma unidade o número de cores por vértice a cada iteração, até que o valor da coloração fracionária não seja mais melhorado ou que o número de classes de cor atinja um limite determinado N_{lim}.

A variável $\mathcal{C}$ representa a coleção das classes de cor geradas nas iterações do algoritmo. O operador $\uplus$ representa a união de coleções, podendo haver repetições de elementos (no nosso caso, classes de cor) na coleção resultante ($\{C\} \uplus \{C\} = \{C, C\} \neq \{C\}$). Toda classe de cor gerada em uma iteração pode ser estendida em qualquer iteração posterior a iteração corrente. Para cada iteração k, a variável U indica o conjunto de vértices ainda não coloridos durante a iteração k, e a variável t contém o valor da última coloração fracionária obtida.

Utilizamos o *DSATUR* como heurística de coloração inteira na linha 8. Para ilustrar o funcionamento do Algoritmo 2, considere o exemplo da Figura 8.

Na iteração $k = 1$, $U_1 = \{1, ..., 9\}$ e a primeira coloração gerada é $\mathcal{C}' = \{S_1, ..., S_5\}$,

Algoritmo 2: Algoritmo para obtenção de uma coloração fracionária.

Chamada: FCP($G = (V, E)$)
Entrada: Grafo G.
Resultado: Coloração fracionária.

```
1  Inicializar: t ← ∞, C ← ∅, y ← 0, U ← V, c ← 0, k ← 1
2  repita
3      U ← V
4      para todo v ∈ U faça
5          se existe S ∈ C tal que N[v] ∩ S = ∅ então
6              S ← S ∪ {v}
7              U ← U \ {v}
8      Aplicar heurística de coloração inteira ao grafo G[U], sendo C' a coloração
       resultante
9      c ← c + |C'|
10     se c/k ≤ t então
11         t ← c/k
12         C ← C ⊎ C'
13         k ← k + 1
14     senão
15         c ← N_lim + 1
16 até que c > N_lim
17 retorne C
```

com $S_1 = \{1, 9\}$, $S_2 = \{2, 5\}$, $S_3 = \{3, 4\}$, $S_4 = \{6, 8\}$ e $S_5 = \{7\}$. Isso resulta no valor de $t = |\mathcal{C}| = 5$.

Em $k = 2$, o vértice 2 pode ser adicionado à classe de cor S_5 e o vértice 4 à classe de cor S_4, ou seja, $S_4 = \{4, 6, 8\}$ e $S_5 = \{2, 7\}$. Assim, temos $U_2 = \{1, 3, 5, 6, 7, 8, 9\}$. A heurística de coloração inteira gera $\mathcal{C}' = \{S_6, S_7, S_8, S_9\}$, com $S_6 = \{1, 8\}$, $S_7 = \{3, 5, 9\}$, $S_8 = \{6\}$ e $S_9 = \{7\}$. Assim, $(|\mathcal{C}| + |\mathcal{C}'|)/2 = 4.5 < 5$, então $t = 4.5$ e $\mathcal{C} = \mathcal{C} \uplus \mathcal{C}'$.

Para $k = 3$, o vértice 2 pode ser adicionado à classe de cor S_8, os vértices 4 e 8 à classe S_9, ou seja, $S_8 = \{2, 6\}$ e $S_9 = \{4, 7, 8\}$. Então, temos $U_3 = \{1, 3, 5, 6, 7, 9\}$. É gerada a coloração inteira $\mathcal{C}' = \{S_{10}, S_{11}, S_{12}, S_{13}\}$, com $S_{10} = \{1, 9\}$, $S_{11} = \{3, 5\}$, $S_{12} = \{6\}$ e $S_{13} = \{7\}$. Assim, $(|\mathcal{C}| + |\mathcal{C}'|)/3 = 4.33 < 4.5$, então $t = 4.33$ e $\mathcal{C} = \mathcal{C} \uplus \mathcal{C}'$.

Se considerarmos $N_{lim} = 10$, então a iteração 3 é a última e terminamos com $\mathcal{C} = \{S_1, ..., S_{13}\}$ e $t = 4.33$.

A complexidade do *FCP* é determinada pela complexidade de execução das linhas 3 – 7 e linha 8 em cada iteração do algoritmo. Como o número máximo de cores é N_{lim}, a execução das linhas 3 – 7 pode ser feita em $O(|V|^2 \cdot N_{lim})$. Utilizando a heurística *DSATUR*, a execução da linha 8 pode ser feita em $O(|V|^3)$. Como o número máximo de iterações do algoritmo é N_{lim}, pois a cada iteração pelo menos um vértice é colorido, a complexidade do *FCP* é $O(N_{lim} \cdot (|V|^2 \cdot N_{lim} + |V|^3))$ ou $O(|V|^2 \cdot N_{lim}^2 + |V|^3 \cdot N_{lim})$.

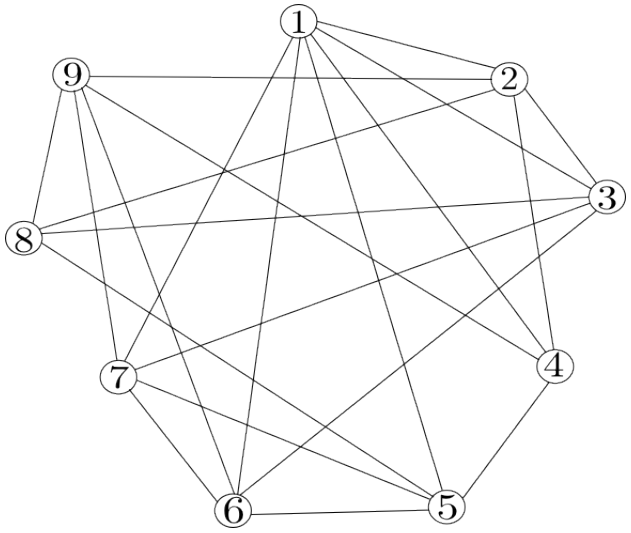

Figura 8 – Exemplo de instância para o problema de *Coloração Fracionária*.

Uma análise dos resultados experimentais da heurística *FCP* apresentada em (BALAS; XUE, 1996) permite concluir que, apesar de essa heurística produzir aproximações do número cromático fracionário de melhor qualidade que o *DSATUR*, essa aproximação ainda é, na maioria dos casos, pior que $\chi_F(G)$.

2.5 Heurística do Grafo Replicado

Elaboramos uma heurística que faz uso de uma solução fracionária e racional oriunda de uma relaxação de alguma formulação de programação inteira (por exemplo, as formulações naturalmente decorrentes das definições de poliedros nas Seções 3.1 e 2.2). A exigência que se faz sobre essa solução fracionária é que seja possível atribuir a cada cor C_i e a cada vértice v o valor fracionário correspondente ao número de vezes que v aparece em C_i no conjunto de cores da coloração (equivalente ao valor fracionário do número de vezes que C_i aparece na coloração). Chamemos esse valor, para cada conjunto independente S de G, de $y_S[v]$.

A primeira etapa do procedimento para transformar y em uma coloração fracionária visa obter o número k de cores que cada vértice de G recebe. Para isso, escreve-se os valores $y_S[v]$ que não sejam nulos na forma irredutível ν_v^S/k_v^S. O *MMC* dos denominadores desses valores em fração irredutível nos dará o número de cores que cada vértice de G recebe.

No Algoritmo 3, é obtida uma coloração k-dimensional para G. Para isso, fazemos uso do grafo k-replicado obtido a partir de G. O grafo k-replicado $G^k = (V^k, E^k)$ de G é definido da seguinte forma:

- Para cada vértice $v \in V$, existe uma clique CL_v em G^k, associada a v, com $|CL_v| = k$

- $V^k = \bigcup_{v \in V} CL_v$
- $(u, v) \in E \Leftrightarrow (u', v') \in E^k, u' \in CL_u, v' \in CL_v$

Uma ilustração da replicação de um grafo é apresentada na Figura 9.

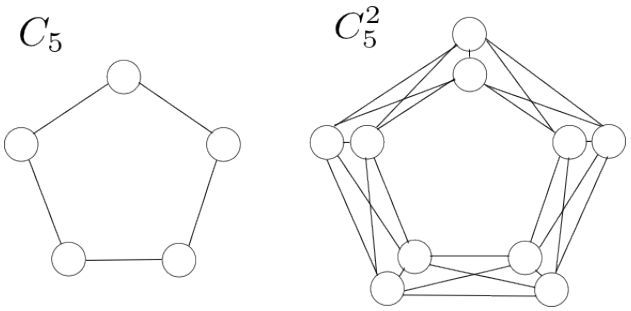

Figura 9 – Ilustração da replicação de um grafo. Grafo C_5, 2-replicado.

Algoritmo 3: Algoritmo para obtenção de uma coloração k-dimensional.

Chamada: KCORES(G, k)
Entrada: Grafo G, valor inteiro positivo k.
Resultado: Coloração k-dimensional.

1 Criar o grafo k-replicado de G, G^k
2 Colorir G^k utilizando heurística de coloração inteira, gerando uma coloração C
3 **retorne** C

Na linha 2 do Algoritmo 3, uma coloração do grafo k-replicado é construída com uma heurística de coloração inteira. Em princípio, utilizamos a heurística *DSATUR*. Porém, verificamos que tal heurística não é muito eficiente para os grafos replicados por conta de sua estrutura. Por isso, usamos a heurística *DSATUR* com algumas modificações na ordem de escolha do vértice a ser colorido.

- O grau de saturação para um vértice $u \in V$ é definido como $\sum_{c \in Cor} CTR(u, c)$, onde $CTR(u, c) = \max\{p(v, u) : Cor(v) = c \text{ e } (v, u) \in E\}$ e

$$p(v, u) = \begin{cases} 0, & \text{se } (v, u) \notin E \text{ ou } (v, u \in \text{clique } CL_z \text{ em } G^k, z \in V) \\ 1, & \text{caso contrário} \end{cases}$$

- Os vértices não coloridos são ordenados pelo maior valor de grau de saturação corrente. No caso de empate, o vértice com menor número de cores em sua clique associada no grafo replicado tem prioridade. No caso do empate persistir, o vértice de maior grau tem prioridade.

As Figuras 10 e 11 mostram o resultado da aplicação do *DSATUR* e do *DSATUR modificado* no grafo replicado da Figura 9, respectivamente. Observe que na heurística *DSATUR*, os vértices associados a uma mesma clique CL_v no grafo replicado são coloridos como se fossem um só vértice, gerando uma coloração fracionária de valor $6/2 = 3$. Porém, na coloração obtida pelo *DSATUR modificado* obtemos o valor $5/2 = 2.5$. Isso se deve ao fato de que a coloração de um vértice $z \in CL_v$ do grafo replicado em uma iteração do *DSATUR modificado* não incrementa os graus de saturação dos vértices pertencentes à mesma clique CL_v associada a $v \in V$.

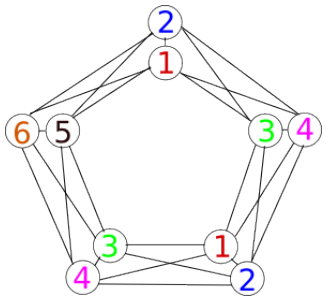

Figura 10 – Ilustração da coloração gerada pelo *DSATUR* aplicado ao grafo replicado da Figura 9

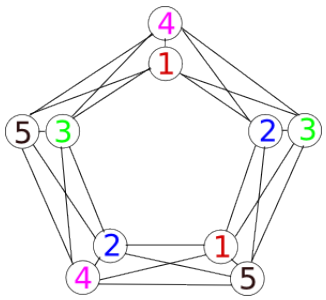

Figura 11 – Ilustração da coloração gerada pelo *DSATUR modificado* aplicado ao grafo replicado da Figura 9

O Algoritmo 4 mostra como gerar uma coloração fracionária a partir das ideias explicadas anteriormente.

O algoritmo é simples, porém não produz um bom resultado comparado a outras heurísticas. Experimentalmente comprovamos que o valor do k calculado é geralmente muito alto e então usamos apenas os dois primeiros dígitos desse valor para que o grafo replicado não se tornasse muito grande.

Como exemplo, os valores de k calculados para as instâncias `myciel`4, `myciel`5, `myciel`6, `myciel`7, `queen`6.6, `queen`8.8, `queen`9.9, `queen`11.11 possuem 17 ou 18 dígitos, uma enorme quantidade de cores.

Essa heurística mostra bem a dificuldade de se obter uma boa coloração fracionária a partir de uma solução fracionária do modelo linear-inteiro.

Algoritmo 4: Algoritmo para obtenção de uma coloração fracionária.

Chamada: ColFrac($G = (V, E)$, y)
Entrada: Grafo G, valores y_S obtidos da solução ótima de uma relaxação.
Resultado: Coloração fracionária.

```
1 para cada v ∈ V faça
2     para cada S conjunto independente de G faça
3         Escrever o número racional y_S[v] ≠ 0 na forma irredutível ν_v^S / k_v^S
4 k ← MMC({k_v^S | v ∈ V, S conjunto independente de G})
5 C ← KCORES(G,k)
6 retorne C
```

2.6 Heurística WFCP

O problema *CFP* pode ser formulado como a seguinte variação do problema *CF* (ver formulação na Seção 2.2):

$$\begin{aligned} \min \quad & \tfrac{1}{k} \textstyle\sum_{S \in \mathcal{S}} \omega'(S) \\ \text{s.a.} \quad & \textstyle\sum_{S \in \mathcal{S}(u)} \omega'(S) \geq k \cdot p(u) \qquad u \in V \\ & \omega'(S) \in \mathbb{N} \qquad\qquad S \in \mathcal{S} \\ & k \in \mathbb{N} \end{aligned} \tag{2.7}$$

onde $p(u)$ é uma ponderação do vértice u que indica a quantidade relativa de cores que u deve receber. Mais especificamente, dados dois vértices $u, v \in V$, o número de cores que u recebe é $p(u)/p(v)$ vezes o número de cores de v. No caso não-ponderado (problema *CF*), temos $p(u) = 1$ fixo, $\forall u \in V$, o que significa que todos os vértices recebem o mesmo número de cores. Da mesma forma que no caso não-ponderado, queremos minimizar a razão do número de cores distintas sobre o número de vezes que cada vértice u recebe $p(u)$ cores.

A heurística *FCP*, mostrada na Seção 2.4, pode ser generalizada para que seja aplicada ao caso ponderado onde $p(u) \in \mathbb{N}$, $\forall u \in V$. Como mostrado em (BALAS; XUE, 1996), a principal modificação visa fazer com que cada vértice u seja colorido, em cada iteração, com uma quantidade suficiente de cores que seja pelo menos o valor de seu peso $p(u)$. Assim sendo, a heurística de coloração inteira utilizada na heurística *FCP* deve ser substituída por uma heurística de coloração inteira ponderada, onde cada vértice pode pertencer a mais de uma classe de cor, que por sua vez pode ser escolhida ou *ponderada* mais de uma vez.

A heurística para o caso ponderado é chamada *WFCP* (*Weighted Fractional Coloring Procedure*) e é apresentada no Algoritmo 5. O algoritmo é novamente iterativo e, para cada

iteração $k \geq 1$, deve-se construir uma coloração ponderada que satisfaça as restrições (2.7) para todos os vértices, supondo que essas mesmas restrições estejam satisfeitas para $k-1$. Para cada classe de cor S_i, $y_{S_i} \in \mathbb{N}$ indica o número de vezes que S_i aparece na coleção de cores da solução. Se $\mathcal{C}$ é uma coleção de classes de cor, então $y(\mathcal{C}) = \sum_{S \in \mathcal{C}} y_S$. De forma a verificar as restrições (2.7) durante a construção da coloração ponderada no Algoritmo 6 na iteração k, para cada $u \in V$, denotamos por $r(u)$ o peso residual que precisa ser "coberto"por classes de cor ponderadas durante a iteração corrente, ou seja, o número de cores necessárias para completar o peso $p(u)$ na iteração corrente.

Algoritmo 5: Algoritmo para obtenção de uma coloração fracionária ponderada em G.

Chamada: WFCP(G, p)
Entrada: Grafo G e ponderação p dos vértices do grafo.
Resultado: Coloração fracionária ponderada por p.

```
1  Inicializar: t ← ∞, C ← ∅, y ← 0, U ← V, c ← 0, k ← 1
2  para todo u ∈ U faça
3      r(u) ← 0
4  repita
5      para cada u ∈ U faça
6          r(u) ← r(u) + p(u)
7      U ← {u ∈ V | r(u) > 0}
8      para cada u ∈ U faça
9          se existe S ∈ C tal que N[u] ∩ S = ∅ então
10             S ← S ∪ {u}
11             r(u) ← r(u) − y_S
12             se r(u) ≤ 0 então
13                 U ← U \ {u}
14     Aplicar heurística de coloração inteira ponderada por r ao grafo G[U], sendo
       (C', y') a coloração resultante
15     c ← c + y'(C')
16     se c/k ≤ t então
17         t ← c/k
18         para todo S ∈ C' faça
19             y_S ← y'_S
20         C ← C ⊎ C'
21         k ← k + 1
22     senão
23         c ← N_lim + 1
24 até que c > N_lim
25 retorne (C, y)
```

Uma execução do Algoritmo 6 inicia-se com $\mathcal{C}_\infty = \emptyset$ e $\ell = 1$. Em seguida, determina-se um conjunto independente S_ℓ incluindo o máximo de vértices de U possível. À ponderação

y_{S_ℓ} de S_ℓ é atribuída o menor peso residual dentre os seus membros, $r_{min} = \min\{r(u) \mid u \in S_\ell\}$. Os pesos residuais dos membros de S_ℓ são também atualizados, retirando de cada um deles o valor r_{min}. Observe que a atualização de y_S faz com que a restrição do tipo (2.7) referente aos vértices de S cujos resíduos atinjam o valor zero passam a estar satisfeitas por y.

Analogamente ao caso não-ponderado, em cada iteração do algoritmo denotamos por U o conjunto de vértices ainda não coloridos totalmente (vértices que possuem peso residual positivo) durante essa iteração, e por t o valor da última solução fracionária ponderada obtida.

Algoritmo 6: Algoritmo para obtenção de uma coloração inteira ponderada.

Chamada: WSCH($G[U]$, r)
Entrada: Grafo $G[U]$ e peso residual r dos vértices do grafo.
Resultado: Coloração inteira ponderada por r.

```
1  C ← ∅
2  repita
3      Crie uma classe de cor S e inclua o máximo de vértices u ∈ U possível, em
       ordem decrescente de peso residual
4      C ← C ⊎ {S}
5      r_min ← min{r(u) | u ∈ S}
6      y_S ← r_min
7      para cada u ∈ S faça
8          r(u) ← r(u) − r_min
9          se r(u) = 0 então
10             U ← U \ {u}
11 até que U = ∅
12 retorne (C, y)
```

Para ilustrar o funcionamento do Algoritmo 5, considere o exemplo da Figura 12, onde os valores nos quadrados indicam os pesos.

Na iteração $k = 1$, $r = p = (3, 3, 2, 2, 2, 1)$, $U_1 = \{1, ..., 6\}$ e a primeira coloração gerada é $\mathcal{C}' = \{S_1, ..., S_5\}$, com $S_1 = \{1, 4\}$, $y_{S_1} = 2$; $S_2 = \{1, 3\}$, $y_{S_2} = 1$; $S_3 = \{2, 5\}$, $y_{S_3} = 2$; $S_4 = \{2, 6\}$, $y_{S_4} = 1$ e $S_5 = \{3\}$, $y_{S_5} = 1$. Isso resulta no valor de $t = y(\mathcal{C}) = y(\mathcal{C}') = 7$.

Em $k = 2$, $r = (3, 3, 2, 2, 2, 1)$ e $U_2 = \{1, ..., 6\}$. O vértice 5 pode ser adicionado à classe de cor S_5, ou seja, $S_5 = \{3, 5\}$, $r(5) = 2 - 1 = 1$. U_2 não se altera, mas $r = (3, 3, 2, 2, 1, 1)$. A heurística de coloração inteira ponderada gera $\mathcal{C}' = \{S_6, S_7, S_8, S_9\}$, com $S_6 = \{1, 3\}$, $y_{S_6} = 2$; $S_7 = \{1, 6\}$, $y_{S_7} = 1$; $S_8 = \{2, 5\}$, $y_{S_8} = 1$ e $S_9 = \{2, 4\}$, $y_{S_9} = 2$. Assim, $(y(C) + y(\mathcal{C}'))/2 = (7 + 6)/2 = 6.5 < 7$, então $t = 6.5$ e $\mathcal{C} = \mathcal{C} \uplus \mathcal{C}'$.

Para $k = 3$, $r = (3, 3, 2, 2, 2, 1)$ e $U_2 = \{1, ..., 6\}$. As classes de cor em $\mathcal{C}$ são todas maximais. A aplicação da heurística *WSCH* nos retorna uma coloração $\mathcal{C}' = \{S_{10}, S_{11}, S_{12}, S_{13}, S_{14}\}$, com $S_{10} = \{1, 4\}$, $y_{S_{10}} = 2$; $S_{11} = \{1, 3\}$, $y_{S_{11}} = 1$; $S_{12} = \{2, 5\}$, $y_{S_{12}} = 2$; $S_{13} = \{2, 6\}$, $y_{S_{13}} = 1$ e $S_{14} = \{3\}$, $y_{S_{14}} = 1$. Assim, $(y(C) + y(\mathcal{C}'))/3 =$

$(13 + 7)/3 = 6.67 > 6.65$, então paramos o algoritmo com os valores de variáveis $t = 6.5$, $\mathcal{C} = \{S_1, ..., S_9\}$, $y_{S_1} = 2$, $y_{S_2} = 1$, $y_{S_3} = 2$, $y_{S_4} = 1$, $y_{S_5} = 1$, $y_{S_6} = 2$, $y_{S_7} = 1$, $y_{S_8} = 1$ e $y_{S_9} = 2$.

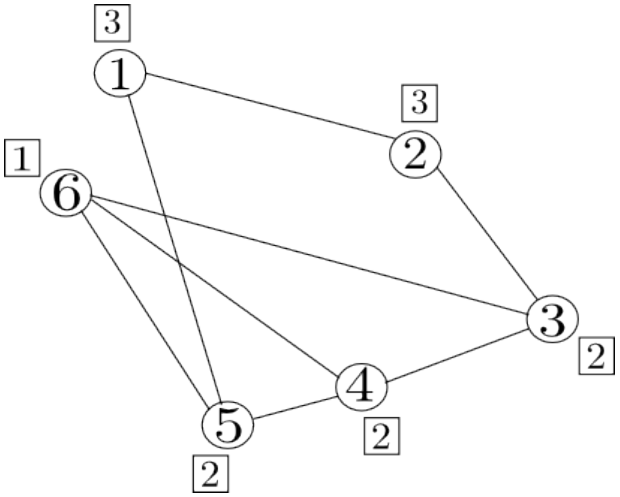

Figura 12 – Exemplo de instância para o problema de *Coloração Fracionária Ponderada*.

3 Uma Heurística Lagrangeana para o Problema de Coloração Fracionária

Neste capítulo, propomos uma heurística lagrangeana para o problema *CF*. A heurística é baseada em uma relaxação lagrangeana de uma formulação para o problema diferente da formulação apresentada no Capítulo 2. A formulação adotada é a formulação por representantes, inicialmente proposta em (CAMPêLO; CAMPOS; CORRêA, 2009), e apresentada ao longo deste capítulo. Contrariamente à formulação do Capítulo 2, a formulação por representantes envolve um número polinomial (no número de vértices do grafo) de variáveis. Por essa razão, a técnica de geração de colunas perde seu interesse. Essa formulação possui duas características exploradas na heurística lagrangeana aqui proposta. A primeira dessas características é o fato de conter restrições do poliedro de conjuntos independentes, o que nos permite usar técnicas já conhecidas de separação de várias das suas facetas. A segunda característica que exploramos é uma decomposição da formulação de relaxação lagrangeana em subproblemas independentes, permitindo assim a resolução dos mesmos em paralelo.

Primeiro apresentamos a notação da modelagem por representantes e uma formulação segundo essa modelagem para o problema *CF* na Seção 3.1. Na Seção 3.2, mostramos a relaxação lagrangeana da formulação utilizada para nossa heurística lagrangeana. Um método para encontrar boas soluções viáveis para o problema fazendo uso de uma heurística lagrangeana é apresentado na Seção 3.3. A heurística lagrangeana usada nesse método é mostrada na seção seguinte. Por fim, na Seção 3.5 propomos uma maneira de acelerar a execução do método da Seção 3.3.

3.1 Modelagem por Representantes

Iniciamos a apresentação da modelagem pelo problema *CI* para, em seguida, apresentar a sua adaptação para o problema *CF*. Nessa apresentação, o grafo $G = (V, E)$ é um grafo simples (sem arestas múltiplas) tal que $V = [n] = \{0, 1, \ldots, n-1\}$, e alguma notação adicional é necessária.

3.1.1 Notação

Ao longo do texto, usamos a seguinte notação para modelagem por representantes de problemas de programação linear inteira. Seja O *complemento* de G é denotado por $\bar{G} = (V, \bar{E})$. A *vizinhança* e a *anti-vizinhança* de um vértice $u \in V$ são dadas por $N(u) = \{v \in V \mid uv \in E\}$ e $\bar{N}(u) = V \backslash (N(u) \cup \{u\})$, respectivamente. Definimos a *anti-*

vizinhança positiva de u por $\bar{N}^+(u) = \{v \in \bar{N}(u) \mid u < v\}$ e a *anti-vizinhança negativa* como $\bar{N}^-(u) = \{v \in \bar{N}(u) \mid v < u\}$. Além disso, denotamos $\bar{N}^+[u] = N^+(u) \cup \{u\}$ e $\bar{N}^-[u] = N^-(u) \cup \{u\}$.

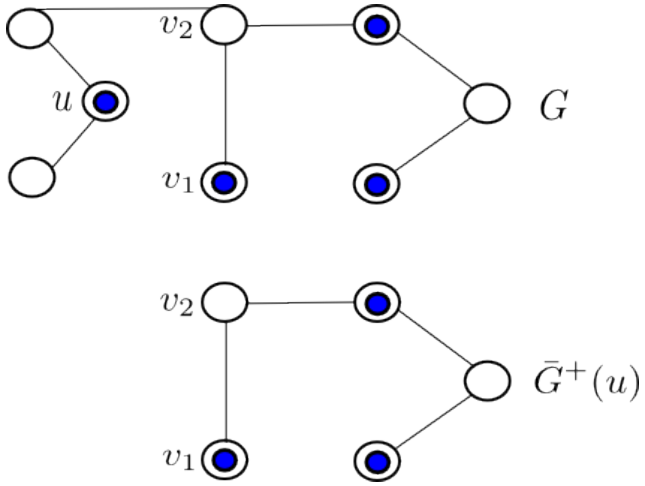

Figura 13 – Exemplo de notação para a modelagem por representantes. Supomos que $u < v_1$ e $u < v_2$.

Seja $H \subseteq V$, então $\bar{G}[H] = (H, \bar{E}[H])$ é o subgrafo induzido em $\bar{G}$ por H, cujos vértices são os elementos de H e cujas arestas são aquelas de $\bar{G}$ com ambas as extremidades em H. Quando H é tal que $\bar{G}[H]$ possui um conjunto vazio de arestas, H é chamado de *conjunto independente* de $\bar{G}$. Usamos $\bar{G}^-(u)$ para representar $G[\bar{N}^-(u)]$ e $\bar{G}^+(u)$ para representar $G[\bar{N}^+(u)]$. Da mesma forma, $\bar{G}^-[u] = G[\bar{N}^-[u]]$ e $\bar{G}^+[u] = G[\bar{N}^+[u]]$. A Figura 13 ilustra a notação para um grafo G. Nesse exemplo, supomos que $u < v_1$ e $u < v_2$.

3.1.2 Problema CI

Uma coloração pode ser descrita como uma família de conjuntos independentes, cada um formado pelos vértices que recebem mesma cor da coloração. Na modelagem por representantes, cada cor da coloração é expressa por um vértice representante que é assim chamado por representar os demais vértices do conjunto independente associado a essa cor.

Nesse sentido, definimos um vetor $x^u \in \{0,1\}^{|\bar{N}^+[u]|}$, para todo $u \in V$, que contém variáveis binárias indexadas pelos vértices em $\bar{N}^+[u]$. Para $u, v \in V$, $u \leq v$ e $uv \in \bar{E}$, $x^u[v] = 1$ se, e somente se, u representa v (mais precisamente, u representa a cor atribuída a v). Além disso, $x^u[u] = 1$ se, e somente se, u é representante de alguma cor. Dessa forma, as restrições referentes a uma coloração inteira em G são as seguintes:

$$\sum_{u \in \bar{N}^-[v]} x^u[v] \geq 1 \qquad v \in V \tag{3.1}$$

$$x^u[v] + x^u[w] \leq x^u[u] \qquad u \in V, vw \in E, v, w \in \bar{N}^+(u) \tag{3.2}$$

$$x^u[v] \in \{0,1\} \qquad u \in V, v \in \bar{N}^+[u] \tag{3.3}$$

As restrições (3.1) garantem que cada vértice receba pelo menos uma cor enquanto as restrições (3.2) impedem que dois vértices adjacentes recebam a mesma cor. Chamamos as restrições (3.2) de *restrições de aresta*. O vetor característico $x^u \subseteq \{0,1\}^{|\bar{N}^+(u)|}$ de um conjunto independente $S \subseteq V$ representado por u no caso em que $x^u[u] = 1$ é, então, um vetor que satisfaz as restrições (3.2) e (3.3), onde $x^u[v] = 1$, $v \in \bar{N}^+(u)$ se, e somente se, $v \in S$. No caso $x^u[u] = 0$, o conjunto independente representado por u é vazio. A Figura 14 ilustra a notação das variáveis de x para o exemplo da Figura 13. Os vértices com marcação azul correspondem aos vértices do conjunto independente representado por u.

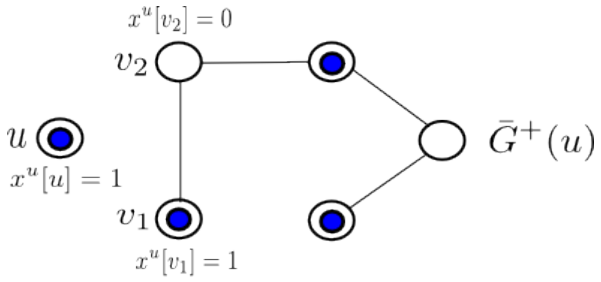

Figura 14 – Exemplo de notação para a modelagem por representantes. Os vértices com marcação azul correspondem aos vértices do conjunto independente representado por u.

3.1.3 Problema CF

Iniciamos a apresentação da formulação para o problema *CF* com uma adaptação das restrições (3.1) – (3.3) que levam a uma relaxação do conjunto de colorações fracionárias de G:

$$\sum_{u \in \bar{N}^-[v]} x^u[v] \geq k \qquad v \in V \tag{3.4}$$

$$x^u[v] + x^u[w] \leq x^u[u] \qquad u \in V, vw \in E \mid v, w \in \bar{N}^+(u) \tag{3.5}$$

$$x^u[u] \leq k \qquad u \in V \tag{3.6}$$

$$x^u[v] \in \mathbb{N} \qquad u \in V, v \in \bar{N}^+[u] \tag{3.7}$$

$$k \in \mathbb{N} \tag{3.8}$$

Aplicando essas restrições, dados vértices u e v, a variável $x^u[u]$ indica o número de conjuntos independentes representados por u (pode haver repetição de conjuntos independentes) e a variável $x^u[v]$ indica o número de conjuntos independentes representados por u que incluem o vértice v. Além disso, a variável k indica o número mínimo de cores que cada vértice deve receber.

As restrições mostradas acima ainda não são suficientes para descrever uma coloração fracionária, razão pela qual mencionamos ser uma relaxação. A imprecisão da formulação está no fato de as restrições (3.5) não serem mais suficientes para descrever conjuntos independentes. Por exemplo, considere um vértice u e uma clique (v,w,z) na sua anti-vizinhança positiva. A solução $x^u[u] = 2$, $x^u[v] = 1$, $x^u[w] = 1$ e $x^u[z] = 1$ satisfaz a restrição (3.5), porém indica que cada um dos vértices da clique está em metade dos conjuntos independentes representados por u. Consequentemente, em pelo menos um desses conjuntos há dois vértices da clique. Assim sendo, é preciso adicionar outras restrições (restrições de cliques maximais, buracos, anti-buracos, entre outras) para descrever de maneira correta uma formulação por representantes para restrições de coloração fracionária.

Seja

$$\mathcal{Y}(G) = \{y \in \{0,1\}^{|V|} \mid y[v] + y[w] \leq 1,\ v, w \in V, vw \in E\}$$

o conjunto dos vetores característicos dos conjuntos independentes em G. Considerando um valor $s \in \mathbb{N}$ fixo, definimos também o conjunto

$$STAB^I_s(G) = \{z \in \mathbb{N}^{|V|} \mid z = \sum_i \alpha_i y_i,\ \sum_i \alpha_i = s,$$
$$\alpha_i \in \mathbb{N}, y_i \in \mathcal{Y}(G), 1 \leq i \leq |\mathcal{Y}(G)|\}.$$

Observe que os coeficientes $\alpha_1, \ldots, \alpha_{|\mathcal{Y}(G)|}$ definem uma combinação linear de conjuntos independentes de G. Levando em conta que todos esses coeficientes são números naturais e que sua soma é s, logo $STAB^I_s(G)$ é o conjunto de todas as possibilidades de escolha de s conjuntos independentes em G. As restrições (3.4), (3.6) e (3.8) acrescidas de $x^u \in STAB^I_{x^u[u]}(G)$, para cada $u \in V$, representam exatamente o poliedro de coloração multidimensionais (CAMPêLO; CAMPOS; CORRêA, 2009).

Realizando uma transformação de variáveis de forma tal que a variável $x^u[v]$ passe a ter o significado de $x^u[v]/k$ em (3.4) – (3.8), para todo $u \in V$ e $v \in \bar{N}^+[u]$, chegamos à seguinte formulação:

$$\sum_{u \in \bar{N}^-[v]} x^u[v] \geq 1 \qquad v \in V \tag{3.9}$$

$$x^u \in STAB_{x^u[u]}(\bar{G}^+(u)) \qquad u \in V \tag{3.10}$$

$$x^u[u] \leq 1 \qquad u \in V \tag{3.11}$$

$$x^u[v] \in \mathbb{Q}_+ \qquad u \in V, v \in \bar{N}^+[u] \tag{3.12}$$

Para essa transformação, dado $s \in \mathbb{Q}_+$ fixo, o conjunto de todas as possibilidades de escolha de s conjuntos independentes se torna

$$STAB_s(G) = \{z \in \mathbb{Q}_+^{|V|} \mid z = \sum_i \alpha_i y_i,\ \sum_i \alpha_i = s, \alpha_i \in \mathbb{Q}_+, \\ y_i \in \mathcal{Y}(G), 1 \leq i \in \mathbb{N} \leq |\mathcal{Y}(G)|\}.$$

As restrições (3.9) – (3.12) representam exatamente o poliedro de coloração fracionária. Realizando a transformação de variáveis $\beta_i = \frac{\alpha_i}{s}$, para todo $u \in V$, $1 \leq i \in \mathbb{N} \leq |\mathcal{Y}(G)|$, temos $STAB_s(G) = \{z \in \mathbb{R}_+^{|V|} \mid z = \sum_i s\beta_i y_i,\ \sum_i \beta_i = 1, y_i \in \mathcal{Y}(G), 1 \leq i \in \mathbb{N} \leq |\mathcal{Y}(G)|\}$. Chegamos então à expressão:

$$STAB_s(G) = conv\{y \in \{0, s\}^{|V|} \mid y[v] + y[w] \leq s,\ v, w \in V, vw \in E\} \tag{3.13}$$

No caso em que $s = 1$, denotamos apenas por $STAB(G)$. Observe que existe uma relação biunívoca entre $STAB_s(G)$ e $STAB(G)$ na qual todo elemento de $STAB_s(G)$ corresponde a um elemento de $STAB(G)$ multiplicado por s.

A formulação (3.9) – (3.12) envolve um número de variáveis polinomial, $|V| + |\bar{E}|$, em detrimento da quantidade potencialmente exponencial de restrições. Todas as restrições de racionalidade que aparecem podem ser eliminadas, como explicado na formulação por conjuntos independentes da Seção 2.2. A transformação para formulação linear elimina a única variável inteira k com o intuito de facilitar a resolução do problema. Porém, neste trabalho, queremos também a coloração fracionária que resulta no valor ótimo. Assim sendo, se usássemos essa formulação para resolvermos o problema, deveríamos transformar a solução desta formulação linear em uma solução da formulação não-linear onde temos o valor de k. Essa não é uma tarefa fácil, visto que precisamos encontrar um valor de k que torne, para cada $u \in V$, $\alpha_i = k \cdot x^u[u] \cdot \beta_i$ inteiro, para todo $1 \leq i \in \mathbb{N} \leq |\mathcal{Y}(G)|$. Por isso, como veremos na Seção 3.2, as formulações implementadas neste trabalho são usadas apenas para gerar limites inferiores e certas informações para heurísticas, e então não nos preocupamos com essa questão, pois usamos heurísticas para determinar a solução viável.

3.2 Relaxação Lagrangeana para a Formulação por Representantes do Problema CF

O método da relaxação lagrangeana consiste em remover algumas das restrições da formulação original e inseri-las na função objetivo. Para conseguir uma boa relaxação dessa forma, isto é, uma relaxação cujo valor ótimo nos retorna um bom limite dual, são usados penalizadores. Cada restrição removida possui um penalizador (coeficiente) associado, que combinados controlam a violação de tais restrições pela função objetivo. Chamamos tais coeficientes de *multiplicadores lagrangeanos*. Para detalhes sobre o método genérico, sugerimos a leitura de (FISHER, 1981).

No restante desta seção, apresentamos uma relaxação lagrangeana da formulação (3.9) – (3.12) para o problema *CF*, com a função objetivo dada por

$$\min \sum_{u \in V} x^u[u]. \tag{3.14}$$

Para tornar independentes os subproblemas definidos para cada vértice u de G no grafo induzido por sua anti-vizinhança positiva, $\bar{G}^+(u)$, as restrições relaxadas são as de (3.9). Observe que tais restrições acoplam variáveis de diferentes representantes, e, ao removê-las, conseguimos separar o problema em um subproblema para cada representante. Mais precisamente, a relaxação do problema para um vetor de multiplicadores λ fixo, $\lambda \geq \mathbf{0}$, se torna:

$$\begin{aligned} L(\lambda) = \quad & \min \ \textstyle\sum_{u \in V} x^u[u] + \lambda[u](1 - \sum_{v \in \bar{N}^-[u]} x^v[u]) \\ & \text{s.a.} \qquad x^u \in STAB_{x^u[u]}(\bar{G}^+(u)), \qquad u \in V \\ & \qquad\quad 0 \leq x^u[u] \leq 1, \qquad u \in V \end{aligned}$$

onde $\lambda[u]$ é o multiplicador associado à restrição relaxada (3.9) referente a u. Em (FISHER, 1981) pode ser encontrada a argumentação para o fato de $L(\lambda)$ ser um limite inferior para o número cromático fracionário de G, para qualquer $\lambda \geq \mathbf{0}$. Observando a função objetivo, constatamos que seriam desejáveis multiplicadores de valor pequeno a fim de maximizar o limite inferior $L(\lambda)$ obtido. Porém, valores pequenos dos multiplicadores lagrangeanos podem não garantir que a solução ótima (ou seja, que define o valor de $L(\lambda)$) seja viável para (3.9) – (3.12).

Reorganizando a função objetivo, chegamos ao problema $FC(\lambda)$, definido como:

$$L(\lambda) = \quad \min \quad \sum_{u\in V} \lambda[u] + (1-\lambda[u])x^u[u] - \sum_{v\in \bar{N}^+(u)} \lambda[v]x^u[v] \tag{3.15}$$

$$\text{s.a.} \qquad x^u \in STAB_{x^u[u]}(\bar{G}^+(u)), \qquad u \in V \tag{3.16}$$

$$0 \leq x^u[u] \leq 1, \qquad u \in V \tag{3.17}$$

Analisando a função objetivo (3.15), observamos que, para cada $u \in V$, a restrição (3.17) pode ser substituída por $x^u[u] \in \{0,1\}$ pois se $(1-\lambda[u]) \geq \max_{\text{(3.16) válido}} \sum_{v\in \bar{N}^+(u)} \lambda[v]x^u[v]$, então o ótimo de (3.15) é atingido em uma solução com $x^u[u] = 0$ e, caso contrário, $x^u[u] = 1$ resulta em um valor da função objetivo melhor ou igual a $x^u[u] = \epsilon$, $0 < \epsilon < 1$.

Seja $f_u = 1 - \sum_{v\in \bar{N}^-[u]} \bar{x}^v[u]$ o valor que representa o quanto a restrição relaxada referente a u é violada pela solução $\bar{x}$ em $FC(\lambda)$. Observe que quanto maior f_u, maior será sua contribuição na função objetivo pois os multiplicadores são não-negativos e, como queremos minimizá-la, a solução ótima de $FC(\lambda)$ tenderá a não violar tal restrição. No caso particular em que $\bar{x}$ é uma solução ótima de $FC(\lambda)$, os valores de $1 - \sum_{v\in \bar{N}^-[u]} \bar{x}^v[u]$, $u \in V$, e de λ estão relacionados na forma de um problema dual descrito a seguir.

Considere o subproblema $FC_u(\lambda)$ de *conjunto independente máximo ponderado* associado a u, para cada $u \in V$:

$$\alpha_u(\lambda) = \quad \max \quad \sum_{v\in \bar{N}^+(u)} \lambda[v]\hat{x}^u[v]$$

$$\text{s.a.} \quad \hat{x}^u \in STAB(\bar{G}^+(u)) \tag{3.18}$$

Seja $Rep(\lambda) = \{u \in V \mid \alpha_u(\lambda) > 1 - \lambda[u]\}$. $Rep(\lambda)$ representa o conjunto de vértices de G que são representantes em uma solução ótima, visto que a atribuição do valor 1 a $x^u[u]$, para um vértice u tal que $\alpha_u(\lambda) > 1 - \lambda[u]$, resulta em um valor da função objetivo (3.15) melhor ou igual ao valor obtido caso tenhamos $x^u[u] = 0$.

Para resolver $FC(\lambda)$, basta resolver os subproblemas $FC_u(\lambda)$, $u \in V$, e, de acordo com a solução ótima $\hat{x}$, construimos a seguinte solução ótima de $FC(\lambda)$:

$$x^u[v] = \begin{cases} 1, & \text{se } u \in Rep(\lambda), v = u \\ \hat{x}^u[v], & \text{se } u \in Rep(\lambda), v \neq u \\ 0, & \text{caso contrário} \end{cases} \qquad u \in V, v \in \bar{N}^+[u]$$

Então, o valor ótimo de $FC(\lambda)$ é igual a

$$\sum_{u\in V\setminus Rep(\lambda)} \lambda[u] + \sum_{u\in Rep(\lambda)} (1 - \alpha_u(\lambda))$$

Por definição, o *Lagrangeano Dual* do problema (3.9) – (3.12) com função objetivo (3.14), que denotamos por (LD), é:

$$(LD) \quad L(\lambda^*) = \begin{array}{ll} \max & L(\lambda) \\ \text{s.a.} & \lambda \in \mathbb{R}_+^{|V|} \end{array}$$

3.3 Método de Duas Fases para o Problema LD

O método que adotamos na elaboração da heurística lagrangeana para o problema *CF* é o chamado *Método do Subgradiente de Duas Fases* (*MSDF*). Ele é baseado no método de três fases proposto em (CAPRARA; FISCHETTI; TOTH, 1995) e consiste em executar alternadamente duas fases de resolução do problema (LD) até que a melhor solução primal viável encontrada não consiga ser melhorada. Trata-se, portanto, de um método iterativo que tem como elemento essencial o *Método do Subgradiente* (REEVES, 1993).

3.3.1 Método do Subgradiente

Dito de forma genérica, o *Método do Subgradiente* (*MS*) é um método iterativo para resolução aproximada do problema dual proveniente de uma relaxação lagrangeana que, a partir de um valor inicial para os multiplicadores lagrangeanos, obtém, a cada iteração, novos valores atualizando-os segundo direções definidas pelos subgradientes das restrições relaxadas. A seguir, fazemos uma apresentação mais específica desse método para o problema (LD). A fim de diferenciar a notação dos diversos valores de λ produzidos pelo método, escrevemos λ^k para denotar o valor de λ no início de uma iteração $k \geq 0$.

A execução do *MS* se inicia na iteração $k = 0$ com valores para λ^0 escolhidos arbitrariamente. Em particular usamos o valor 1 para todos eles, escolhido de maneira arbitrária. Para qualquer iteração $k \geq 0$, o valor λ^k define o problema $FC(\lambda^k)$. A resolução desse problema produz uma solução ótima $\bar{x}_k$. Para esse ponto $\bar{x}_k$, calcula-se o subgradiente $\Delta(\lambda^k)$ das restrições (3.9) com a seguinte expressão:

$$\Delta(\lambda^k)[u] = 1 - \sum_{v \in \bar{N}^-[u]} \bar{x}_k^v[u], \qquad u \in V. \tag{3.19}$$

Essas são direções de melhoria da função objetivo da relaxação lagrangeana com intuito de encontrar um multiplicador cujo valor ótimo seja o mais próximo do ótimo primal.

A atualização dos multiplicadores lagrangeanos é feita segundo

$$\lambda^{k+1}[i] = \max\{\lambda^k[i] + \psi \cdot \Delta(\lambda^k)[i] \cdot \frac{(UB - L(\lambda^k))}{||\Delta(\lambda^k)||^2},\ 0\}, \qquad i \in \{1, ..., |V|\} \tag{3.20}$$

onde UB é qualquer valor que seja limite superior para o problema (3.9) – (3.12) com função objetivo (3.14) e ψ é o valor do passo corrente na direção do subgradiente. Para melhorar a convergência, podemos utilizar como limite superior UB o valor da função objetivo para a solução $\bar{x}_k$ ($c\bar{x}_k$), retornada pela heurística lagrangeana com parâmetro λ^k.

Após algumas iterações, se não obtivermos melhora no valor da solução relaxada, pode ser que o tamanho do passo que estamos dando na direção do subgradiente a cada iteração esteja muito grande. Neste caso, diminuimos gradativamente o passo, até que eventualmente este fique menor que um tamanho mínimo. Se em N_{parado} iterações consecutivas não conseguimos melhorar o melhor limite inferior, $bestLB$, então reduzimos à metade o valor do passo. O procedimento termina quando o valor do passo se torna menor que um parâmetro denotado por S_ψ. A escolha dos parâmetros N_{parado}, S_ψ e ψ inicial pode influenciar bastante a qualidade das soluções.

3.3.2 Método do Subgradiente de Duas Fases

Em nosso trabalho, além de bons multiplicadores lagrangeanos, que determinam bons limites inferiores para o problema, queremos, principalmente, uma boa solução viável. Se um conjunto de multiplicadores λ^1 gera um limite inferior melhor do que outro conjunto de multiplicadores λ^2, não podemos afirmar que a solução viável gerada a partir de uma heurística lagrangeana usando λ^1 é melhor que a solução viável gerada usando λ^2. Mesmo assim, bons multiplicadores lagrangeanos têm maior potencial de gerar boas soluções viáveis quando usados em uma heurística lagrangeana (CAPRARA; FISCHETTI; TOTH, 1995). Por isso, nossa estratégia é gerar vários conjuntos de bons multiplicadores lagrangeanos e, para cada um deles, aplicar uma heurística lagrangeana.

O *MSDF* consiste de duas fases, descritas com mais detalhes nas Subseções 3.3.2.1 e 3.3.2.2, respectivamente. A primeira fase é denominada *Fase do Subgradiente*. Nessa fase, o objetivo é encontrar um conjunto de bons multiplicadores lagrangeanos através do *MS*. A segunda fase é a *Fase da Heurística*, onde uma sequência de conjuntos de bons multiplicadores é determinada e cada conjunto de multiplicadores é passado como entrada para uma heurística lagrangeana gerando uma solução viável para o problema *CF* para possivelmente atualizar a melhor solução encontrada. A estrutura do *MSDF* é descrita pelo Algoritmo 7 e as fases são definidas abaixo:

3.3.2.1 Fase 1 - Fase do Subgradiente

Na primeira fase, realizamos o *MS* com limite superior UB fixo. Inicialmente, na primeira execução da fase 1, usamos UB com valor igual ao valor da função objetivo (3.14) aplicada à solução retornada pela heurística *FCP* (Algoritmo 2) aplicada ao grafo G.

Algoritmo 7: Algoritmo genérico para obtenção de uma boa solução viável.

Chamada: DuasFases(P)
Entrada: Problema P.
Resultado: Boa solução viável para o problema P.

1 **repita**
2 | **Fase do Subgradiente**: encontrar um conjunto de bons multiplicadores lagrangeanos λ^*
3 | **Fase da Heurística**: iniciando com λ^*, gerar uma sequência de conjuntos de bons multiplicadores lagrangeanos e, para cada um deles, aplicar uma heurística lagrangeana para gerar uma solução para P, atualizando a melhor solução encontrada x^*
4 **até *que*** x^* *não possa ser melhorado*
5 **retorne** x^*

Os multiplicadores lagrangeanos são iniciados com os mesmos valores dos multiplicadores que resultaram no melhor limite inferior, $bestLB_{f1,2}$, obtido na iteração anterior do *MSDF*, porém com uma pequena perturbação. Essa perturbação é feita adicionando a cada multiplicador lagrangeano um valor aleatório $\frac{0.1}{rand}$, onde $rand \neq 0$ é um inteiro escolhido aleatoriamente no intervalo $[-100, 100]$.

A fase 1 termina quando o valor de ψ se torna menor que um valor fixado S_ψ.

3.3.2.2 Fase 2 - Fase da Heurística

Nesta fase, o mesmo *Método do Subgradiente* é utilizado, mas com algumas modificações. Os multiplicadores lagrangeanos são iniciados com os mesmos valores dos multiplicadores que resultaram no melhor limite inferior, $bestLB_{f1}$, obtido na primeira fase da iteração corrente do *MSDF*.

O valor do passo ψ é fixo e o limite superior UB, utilizado em (3.20), em cada iteração dessa fase é dado pela aplicação da função objetivo (3.14) na solução retornada pela heurística que desenvolvemos (definida na seção seguinte). A heurística faz uso dos multiplicadores lagrangeanos para determinar uma coloração fracionária.

A fase 2 termina quando N_{f2} iterações são executadas. O valor da melhor solução viável é usado como limite superior fixo na fase 1 da iteração seguinte do *MSDF*.

O *MSDF* para o problema *LD* é mostrado no Algoritmo 8. A função de cada variável é explicada a seguir:

- k: número da iteração corrente na execução de uma determinada fase
- ψ: passo a ser dado na direção do subgradiente
- λ^*: conjunto de multiplicadores cujo limite dual associado é o melhor encontrado na iteração corrente do *MSDF*

- λ^k: conjunto de multiplicadores corrente na execução de uma determinada fase
- $bestLB$: melhor limite dual encontrado durante todo o *MSDF*
- $bestUB$: melhor valor de solução viável para o problema *CF* encontrado durante todo o *MSDF*
- $bestLB_{f1}$: melhor limite dual encontrado durante a última execução da fase do subgradiente
- $bestLB_{f1,2}$: melhor limite dual encontrado durante a última iteração do *MSDF*
- x^*: melhor solução viável encontrada para o problema *CF* durante todo o *MSDF*

As linhas 4 – 19 são referentes a execução da fase do subgradiente. A cada iteração k dessa fase, o problema relaxado é resolvido gerando uma solução de valor $L(\lambda^k)$. De acordo com esse valor, é verificado a necessidade da atualização das variáveis $bestLB$, $bestLB_{f1}$ e λ^*. Se o limite $bestLB_{f1}$ não é atualizado em N_{parado} iterações consecutivas, então o passo é reduzido à sua metade. Em seguida, o subgradiente é calculado e os multiplicadores lagrangeanos são atualizados. A execução da fase 1 termina quando o valor de ψ se torna menor que um valor fixado S_ψ.

A execução da fase da heurística é dada pelas linhas 21 – 35. Ela é similar à execução da fase do subgradiente. Porém, em cada iteração k, o conjunto de multiplicadores corrente λ^k é passado como entrada para uma heurística lagrangeana gerando uma solução x_k. Se o valor cx_k dessa solução é o melhor valor encontrado até o momento, então são atualizadas as variáveis $bestUB$ e a melhor solução encontrada x^*. Além disso, o número de iterações da fase da heurística é fixo e igual a N_{f2}.

As fases do subgradiente e da heurística são executadas alternadamente até que a melhor solução encontrada x^* não possa mais ser melhorada.

3.4 Heurística de Construção de Coloração Fracionária

A heurística que desenvolvemos para coloração fracionária é uma adaptação da heurística *FCP*, alterando a linha 8 do Algoritmo 2. Essa alteração consiste em empregar uma heurística de coloração inteira guiada pelos multiplicadores lagrangeanos. Denominamos de FCPRelLag(G,λ) a heurística de coloração fracionária decorrente, cuja ideia é determinar uma coloração fracionária (representada por um $\bar{x}$ satisfazendo (3.9) – (3.12)) com o valor não muito distante de $L(\lambda)$, caso exista uma solução com essa propriedade. A entrada é formada por um grafo e um vetor λ de multiplicadores, a serem usados como ponderação dos vértices.

A heurística de coloração inteira empregada em *FCPRelLag*, denominada ColRelLag($G[R]$,λ), $R \subseteq V$, é uma heurística gulosa que produz uma coloração de

Algoritmo 8: Algoritmo para obtenção de uma boa solução viável para o problema *CF*.

Chamada: DuasFases(*CF*,$FC(\lambda)$)
Entrada: Problema *CF* e problema relaxado $FC(\lambda)$.
Resultado: Boa solução viável para o problema *CF*.

```
1  λ* ← 1, bestLB ← −∞, bestUB ← ∞, UB ← Heurística(CF)
2  repita
3      k ← 1, ψ ← 2, bestLB_f1 ← −∞, bestLB_f1,2 ← −∞, λ^k ← λ*
4      repita
5          Resolver FC(λ^k) gerando solução x̄_k de valor L(λ^k)
6          se L(λ^k) > bestLB então
7              bestLB ← L(λ^k)
8          se L(λ^k) > bestLB_f1 então
9              bestLB_f1 ← L(λ^k)
10             bestLB_f1,2 ← L(λ^k)
11             λ* ← λ^k
12         senão
13             se k ≥ N_parado então
14                 k ← 1
15                 ψ ← ψ/2
16         Calcular subgradiente Δ(λ^k)
17         Atualizar multiplicadores λ^{k+1} usando UB e L(λ^k)
18         k ← k + 1
19     até que ψ < S_ψ
20     k ← 1, ψ ← 2, λ^k ← λ*
21     repita
22         Resolver FC(λ^k) gerando solução x̄_k de valor L(λ^k)
23         se L(λ^k) > bestLB então
24             bestLB ← L(λ^k)
25         se L(λ^k) > bestLB_f1,2 então
26             bestLB_f1,2 ← L(λ^k)
27             λ* ← λ^k
28         Calcular subgradiente Δ(λ^k)
29         x_k ← HeurísticaLagrangeana(CF,λ^k)
30         se cx_k < bestUB então
31             bestUB ← cx_k
32             x* ← x_k
33         Atualizar multiplicadores λ^{k+1} usando cx_k e L(λ^k)
34         k ← k + 1
35     até que k > N_f2
36     Perturbar o vetor de multiplicadores λ*
37     UB ← bestUB
38 até que bestUB não possa ser melhorado
39 retorne x*
```

$G[R]$. A ordem de coloração dos vértices depende dos multiplicadores lagrangeanos no vetor λ da seguinte forma. Atribuímos, para cada vértice, uma *nota* que determina a ordem de examinação dos mesmos na heurística de coloração inteira. Para $u \in R$, a nota $c_\lambda[u]$ se refere à soma dos coeficientes das variáveis dos vértices que podem representar u. Para cada cor, definimos um representante da mesma forma que no modelo por representantes, ou seja, o menor dos vértices de uma cor, dada a numeração dos vértices de G, é o representante dessa cor. Inicialmente,

$$c_\lambda[u] = 1 - \lambda[u] - \sum_{v \in \bar{N}^-(u) \cap R} \lambda[u]$$

Em cada iteração, um vértice u de menor nota é escolhido para ser colorido. A cor a ele atribuída, digamos c, é a menor dentre as que não estejam presentes na vizinhança de u. Dessa atribuição decorre a atribuição $R \leftarrow R - u$ e uma atualização das notas de outros vértices segundo um dos seguintes dois casos possíveis:

- A cor c possui outros vértices além de u: seja r o menor vértice de cor c. Há dois subcasos:
  - $r = u$: $c_\lambda[v] \leftarrow c_\lambda[v] + \lambda[v]$, para todo v com algum vizinho de cor c cuja nota deixa de receber contribuição de u, $v \in R \cap \bar{N}^+(u)$, e para todo v cuja nota deixa de receber contribuição do menor vértice r' de cor c diferente de u, $v \in R \cap \bar{N}^+(r')$.
  - $r < u$: $c_\lambda[v] \leftarrow c_\lambda[v] + \lambda[v]$, para todo v cuja nota deixa de receber contribuição de u, $v \in R \cap \bar{N}^+(u)$, e para todo v cuja nota deixa de receber contribuição de r, $v \in N(u) \cap R \cap \bar{N}^+(r)$.
- A cor c possui apenas o vértice u: nenhuma nota precisa ser atualizada.

Essa definição das notas é derivada das restrições relaxadas na relaxação lagrangeana, por isso possui informações significativas sobre como encontrar uma boa coloração fracionária. A ideia da atualização das notas é verificar quais variáveis de representação se tornam obrigatoriamente nulas após o vértice da iteração corrente ser colorido, e remover sua contribuição nas notas correspondentes.

No algoritmo *FCPRelLag*, a ordem em que os vértices são examinados na linha 4 do Algoritmo 2 é também determinada pelas notas. Assim sendo, o mesmo cálculo e atualização das notas são feitos nesse ponto do algoritmo, mas com $R = V$. Neste caso, não temos a alternativa de um vértice ser colorido com uma nova cor, pois nesse ponto do algoritmo colorimos apenas os vértices que podem receber cores já existentes.

Para mostrar o funcionamento da heurística *ColRelLag*, considere o exemplo ilustrado pela Figura 15. Para esse exemplo, $R = \{0, 1, 2, 3, 4, 5\}$. Inicialmente, a nota do vértice

Algoritmo 9: Algoritmo para obtenção de uma coloração inteira a partir de uma ordem dos vértices que leva em conta os valores de multiplicadores de Lagrange.

Chamada: ColRelLag($G[R]$, λ)
Entrada: Grafo $G[R]$, vetor de multiplicadores de langrage λ.
Resultado: Coloração inteira.

```
1 para cada u ∈ R faça
2 |  Calcular nota inicial c_λ[u]
3 repita
4 |  Escolher vértice u ∈ R com menor nota c_λ[u]
5 |  Colorir u com a menor cor possível
6 |  R ← R − u
7 |  Atualizar notas c_λ
8 até que R = ∅
9 retorne Coloração
```

5 é dada por $c_\lambda[5] = 1 - 0.05 - \sum_{|\bar{N}^-(5)\cap R|} 0.05 = 0.8$. Realizando o mesmo cálculo para os demais vértices temos $c_\lambda[0] = 0.85$, $c_\lambda[1] = 0.5$, $c_\lambda[2] = 0.78$, $c_\lambda[3] = 0.4$, $c_\lambda[4] = 0.55$ e $c_\lambda[5] = 0.8$. Seja j o número da iteração corrente na execução do algoritmo.

Na iteração $j = 1$, o vértice de menor nota é o vértice 3 e então a primeira cor é criada, $S_1 = \{3\}$.

Na iteração $j = 2$, o vértice de menor nota é o vértice 1, e a menor cor que pode ser atribuída a ele é a cor 1. Portanto, $S_1 = \{3, 1\}$. Após o vértice 1 ser colorido, os vértices 4 e 5 não podem ser mais representados pelo vértice 3, que deixou de ser representante da cor S_1. Assim sendo, $c_\lambda[4] \leftarrow c_\lambda[4] + 0.15$ e $c_\lambda[5] \leftarrow c_\lambda[5] + 0.05$. Temos $c_\lambda[0] = 0.85$, $c_\lambda[2] = 0.78$, $c_\lambda[4] = 0.7$ e $c_\lambda[5] = 0.85$.

Em $j = 3$, o vértice de menor nota é o vértice 4, e a menor cor que pode ser atribuída a ele é a cor 2. Portanto, $S_2 = \{4\}$. Continuamos com $c_\lambda[0] = 0.85$, $c_\lambda[2] = 0.78$ e $c_\lambda[5] = 0.85$.

Para $j = 4$, o vértice 2 possui a menor nota, e a menor cor que pode ser atribuída a ele é a cor 2. Portanto, $S_2 = \{4, 2\}$. Após o vértice 2 ser colorido, o vértice 5 não pode ser mais representado pelo vértice 2, pois possui um vizinho (vértice 4) em S_2. Dessa forma, $c_\lambda[5] \leftarrow c_\lambda[5] + 0.05$. Temos $c_\lambda[0] = 0.85$ e $c_\lambda[5] = 0.9$.

Na iteração $j = 5$, o vértice 0 é colorido com a cor 2, pois $c_\lambda[0] < c_\lambda[5]$, resultando em $S_2 = \{4, 2, 0\}$. Por fim, na última iteração, o vértice 5 é colorido com a cor 1, resultando na coloração final $S_1 = \{3, 1, 5\}$ e $S_2 = \{4, 2, 0\}$.

3.5 Pricing

Em nossos experimentos, verificamos que o tempo de resolução dos subproblemas $FC_u(\lambda)$ em cada iteração do subgradiente pode ser relativamente alto dependendo da

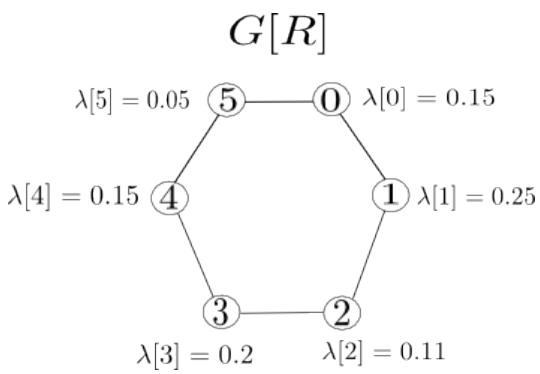

Figura 15 – Exemplo de entrada para a heurística *ColRelLag*. O grafo $G[R]$ é um C_6.

instância. Por isso, implementamos também uma técnica de *pricing* para reduzir o tempo do *MSDF*. A ideia é resolver apenas um subconjunto dos subproblemas em cada iteração do subgradiente. A determinação dos subproblemas que serão resolvidos é feita pelo pricing. Após a execução de um pricing, se um subproblema é determinado inativo, ele não deve ser resolvido até a próxima execução do pricing. Do contrário, se ele é determinado ativo, ele deve ser resolvido. No caso do *MSDF*, o pricing é aplicado apenas na fase 1.

A frequência de aplicação do pricing é comandada por um parâmetro T que representa o número de iterações do subgradiente que serão executadas até que o pricing seja aplicado novamente. Em nossa implementação, inicialmente atribuimos $T = 20$. Depois de cada pricing, calculamos o valor de um parâmetro $\delta = FalseLB - LB$, onde LB é o valor ótimo de $FC(\lambda)$, calculado durante o pricing, e $FalseLB$ é o valor ótimo do problema $(FC_{false}(\lambda))$, calculado antes do pricing, que corresponde ao problema $FC(\lambda)$ com as restrições $x^u[v] = 0$, $v \in \bar{N}^+[u]$, para todo $u \in V$ cujo subproblema $FC_u(\lambda)$ era inativo antes do pricing. Se o parâmetro δ for pequeno, nós diminuimos a frequência de aplicação do pricing aumentando o parâmetro T. Se, ao contrário, δ é grande, então diminuimos o valor de T, pois o valor ótimo de $(FC_{false}(\lambda))$ está longe do valor ótimo de $FC(\lambda)$. Mais precisamente, atualizamos $T = 4 \cdot T$ se $\delta \leq 10^{-5}$; $T = 1.5 \cdot T$ se $\delta \leq 10^{-1}$; $T = 1.1 \cdot T$ se $\delta \leq 0.5$; e $T = 20$ caso contrário. Essa ideia é similar à usada em (CAPRARA; FISCHETTI; TOTH, 1995).

A determinação dos subproblemas ativos e inativos pela aplicação do pricing, executada em uma iteração k, na fase 1 do *MSDF* é feita da seguinte forma:

Seja α_u^k o valor ótimo de $FC_u(\lambda^k)$ para todo $u \in V$ e seja $Rep(\lambda^k) = \{u \in V \mid \alpha_u^k(\lambda^k) > 1 - \lambda^k[u]\}$. $Rep(\lambda^k)$ representa o conjunto de vértices u de G que são representantes em uma solução ótima de $FC(\lambda^k)$, ou seja, $x^u[u] = 1$ resulta em um valor da função objetivo de (3.15) melhor ou igual ao valor obtido fixando $x^u[u] = 0$.

Para cada $u \in V$, se $u \in Rep(\lambda^k)$, então $FC_u(\lambda)$ é ativo, senão ele é inativo. A ideia é tornar inativo os subproblemas cujos vértices associados não são representantes na solução do problema $FC(\lambda^k)$ calculado pelo pricing, pois suspeitamos que tais vértices não

são também representantes na solução ótima de $FC(\lambda^*)$, onde λ^* são os multiplicadores lagrangeanos que geram o melhor limite inferior para o problema CF em G obtidos pelo *MSDF*.

4 Problema de Ponderação de Rodadas

Como mencionado na Seção 1.3, uma solução para o problema *PR* é composta por um fluxo no Grafo da Rede escoando a demanda requerida por cada nó de origem e uma coloração fracionária ponderada pelo fluxo no Grafo de Interferências correspondendo a um protocolo. Para esse problema, G denota o Grafo da Rede e G_I denota o Grafo de Interferências. Neste capítulo, propomos uma heurística lagrangeana para o problema *PR* similar à apresentada para o problema *CF* no Capítulo 3. Portanto, da mesma forma, uma formulação por representantes para o problema *PR* é descrita ao longo do capítulo e uma relaxação lagrangeana dessa formulação é mostrada. Analisando essa relaxação lagrangeana, podemos perceber que sua decomposição gera os mesmos subproblemas de conjunto independente máximo ponderado gerados pela decomposição da relaxação lagrangeana para o problema *CF*, com a adição de um subproblema de fluxo que pode ser facilmente resolvido por um algoritmo combinatório.

Na primeira seção deste capítulo, apresentamos e explicamos as restrições necessárias para a formulação por representantes que é descrita na seção seguinte, 4.2. Na Seção 4.3, mostramos como a heurística *WFCP* pode ser utilizada para determinar uma solução viável para o problema *PR*. Por fim, apresentamos a relaxação lagrangeana da formulação por representantes e a heurística lagrangeana que é utilizada no *MSDF* para o problema *PR* na Seção 4.4.

4.1 Restrições de uma Formulação para o Problema PR

4.1.1 Restrições de Satisfação de Demanda

Para a construção de uma formulação para o problema *PR*, um fluxo $\phi : A \to \mathbb{Q}_+$ é usado para descrever como a demanda gerada pelos nós de origem é escoada até os destinos através da rede. Assim sendo, para cada $uv \in E$, usamos uma variável $\phi[\vec{uv}]$ e uma variável $\phi[\vec{vu}]$ para indicar o fluxo de informação escoado através de cada orientação da aresta do Grafo da Rede em uma execução de um escalonamento do protocolo correspondente a uma solução viável para o problema. Dessa forma, além das restrições de **Conservação de fluxo**, definidas na Seção 1.2, as seguintes restrições também devem ser satisfeitas:

- **Satisfação de demanda**: $\sum_{v \in N(u)} \phi[\vec{uv}] - \phi[\vec{vu}] \geq d(u)$, $\forall u \in V_O$

As restrições de **Satisfação de demanda** impõem que cada origem deva enviar fluxo com quantidade pelo menos o valor da demanda exigida, representando assim a demanda totalmente liberada por tais nós.

Observe que o fluxo em cada orientação de aresta pode ser um valor fracionário. Porém, ele está associado a um fluxo inteiro que precisa ser determinado posteriormente. Como veremos adiante, o fluxo inteiro no período de qualquer escalonamento do protocolo da solução determinada é um múltiplo de ϕ.

4.1.2 Restrições de Coloração Fracionária Ponderada

Sejam ϕ um fluxo que satisfaz as restrições de conservação de fluxo e de satisfação de demanda, $\mathcal{S}$ o conjunto de todos os conjuntos independentes de G_I e $\omega : \mathcal{S} \to \mathbb{Q}_+$ uma ponderação desses conjuntos independentes. A ponderação ω indica o número de vezes que os conjuntos independentes são escolhidos em uma solução em valor fracionário e está associada a um protocolo na rede modelada por G. Essa ponderação pode ser usada como variável em uma formulação para o problema *PR* onde a orientação das transmissões por cada aresta $e \in E$, ativadas em cada rodada que a inclui, é determinada pelo fluxo ϕ. Denotamos por $\mathcal{S}(u,v) = \{S \in \mathcal{S} \mid uv \in S\}$, para $uv \in E$, o conjunto dos conjuntos independentes que incluem a aresta uv. Para que a demanda dos nós de origem seja atendida por um escalonamento do protocolo associado a ω, as restrições a seguir devem ser satisfeitas:

- **Coloração fracionária ponderada**: $\sum_{S \in \mathcal{S}(u,v)} \omega(S) \geq \phi[\vec{uv}] + \phi[\vec{vu}]$, $\forall uv \in E$

4.1.3 Conversão em uma Solução Inteira

O protocolo associado a ω é representado pela ponderação inteira $\omega'(S) = k \cdot \omega(S)$, para todo $S \in \mathcal{S}$, onde $k \in \mathbb{N}$ é o menor valor tal que $\omega'(S) \in \mathbb{N}$, para todo $S \in \mathcal{S}$. Para cada $S \in \mathcal{S}$, $\omega'(S)$ indica o número de vezes que o conjunto independente S aparece no protocolo. Neste caso, a demanda dos nós de origem é atendida pelo menos k vezes por um escalonamento do protocolo definido por ω', no respectivo período, se as seguintes restrições são satisfeitas:

$$\sum_{S \in \mathcal{S}(u,v)} \omega'(S) \geq k(\phi[\vec{uv}] + \phi[\vec{vu}]), \forall uv \in E \tag{4.1}$$

O período de um escalonamento do protocolo definido por ω' é então dado por $\sum_{S \in \mathcal{S}} \omega'(S)$. A razão do período e o número de vezes que a demanda é atendida para cada nó de origem é dada por $\sum_{S \in \mathcal{S}} \omega(S)$. Assim sendo, o problema *PR* pode ser formulado pela determinação de valores às variáveis ω (ponderação dos conjuntos independentes) e ϕ (escoamento da demanda pela rede) que satisfazem as restrições explicadas acima de tal forma a minimizar a expressão $\sum_{S \in \mathcal{S}} \omega(S)$.

4.2 Formulação por Representantes

Uma formulação para o problema deve então combinar as restrições de coloração de arestas do Grafo da Rede e as restrições de fluxo para que a demanda seja atendida pelo protocolo determinado a partir do escoamento proposto pelo fluxo.

Para a descrição de uma formulação por representantes para o problema PR, seguimos a seguinte notação: dada uma aresta $e = uv \in E$ do Grafo da Rede, as orientações de transmissões nessa aresta são denotadas por $\vec{e} = u\vec{v}$ e $\overleftarrow{e} = v\vec{u}$; o fluxo de escoamento da demanda dos nós de origem é representado pelas variáveis $\phi : A \to \mathbb{Q}_+$; a variável $x^{e'}[e]$, $e' \in E$, $e \in \bar{N}^+[e']$, indica a fração do número de conjuntos independentes de G_I representados por e' que incluem e sobre o número total de conjuntos independentes representados por e' em x (pode haver repetição de conjuntos independentes).

$$\min \quad \sum_{e\in E} x^e[e] \tag{4.2}$$

$$\text{s.a.} \quad \sum_{e'\in \bar{N}^-[e]} x^{e'}[e] \geq \phi[\vec{e}] + \phi[\overleftarrow{e}] \qquad e \in E \tag{4.3}$$

$$x^e \in STAB_{x^e[e]}(\bar{G_I}^+(e)) \qquad e \in E \tag{4.4}$$

$$x^e[e] \geq 0 \qquad e \in E \tag{4.5}$$

$$\sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = 0 \qquad u \in V - (V_O \cup V_D) \tag{4.6}$$

$$\sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = d(u) \qquad u \in V_O \tag{4.7}$$

As restrições de coloração são descritas por (4.3) e as de conservação de fluxo por (4.6). As restrições (4.7) definem as restrições de satisfação de demanda. A viabilidade da solução quanto aos conjuntos representados pelas variáveis de representantes serem conjuntos independentes de G_I é garantida pelas restrições (4.4).

Seja $k \in \mathbb{N}$ o número de vezes que a demanda dos nós de origem é atendida, dada uma solução x que satisfaz as restrições da formulação acima. A razão do período pelo valor k é dada por $\sum_{e\in E} x^e[e]$, e a razão do fluxo por período de um escalonamento do protocolo pelo valor k, para $\vec{e}$, $e \in E$, é dada por $\phi[\vec{e}]$. Portanto, a vazão total é dada por $\gamma = \sum_{u\in V_O} \sum_{v\in N(u)} (\phi[u\vec{v}] - \phi[v\vec{u}]) / \sum_{e\in E} x^e[e]$. O problema de maximizar essa função de vazão não é linear. Usando a igualdade (4.7), temos que $\sum_{u\in V_O} \sum_{v\in N(u)} (\phi[u\vec{v}] - \phi[v\vec{u}]) = \sum_{u\in V_O} d(u)$. Dessa forma, podemos transformar a função objetivo em $\min \sum_{e\in E} x^e[e]$. Por (4.3) e pela função objetivo (4.2), podemos concluir que uma solução com $\phi[\vec{e}] > 0$ e $\phi[\overleftarrow{e}] > 0$, para algum $e \in E$, pode sempre ser substituída por uma solução com valor melhor onde o fluxo é nulo para uma das orientações de e. Portanto, para soluções ótimas da formulação, as transmissões ocorrem em apenas uma das orientações em cada aresta. Sendo assim, a solução ótima dada pelas variáveis do vetor x^*, que indicam uma ponderação dos conjuntos independentes de G_I, e ϕ^*, que indica as transmissões

e, consequentemente, as orientações de cada aresta de G, está associada a um protocolo definido.

Para facilitar a apresentação, supomos que $\sum_{u \in V_O} d(u) \leq 1$. Essa suposição pode ser feita sem perda de generalidade pois pode-se proceder a uma normalização das demandas e dos fluxos nos casos em que ela não for satisfeita. Isso pode ser feito a partir da divisão das demandas e de todas as variáveis pelo valor $D = \sum_{u \in V_O} d(u)$. Dessa forma, em uma solução ótima, $\phi[\vec{e}] + \phi[\overleftarrow{e}] \leq 1$, $\forall e \in E$. Juntando isso à hipótese de soma das demandas no máximo unitária e às restrições (4.3) concluimos que existe uma solução ótima com $x^e[e] \leq 1$, $\forall e \in E$. Logo, considerando $x^e[e] \leq 1$ como sendo uma desigualdade válida, o vetor x de uma solução ótima corresponde a uma coloração fracionária (neste caso, ponderada) do Grafo de Interferências G_I (de acordo com a formulação de coloração fracionária descrita na Seção 3.1).

4.3 Heurística WFCP Aplicada ao Problema PR

Suponha que tenhamos determinado um fluxo inteiro inicial $\phi : A \to \mathbb{N}$ no Grafo da Rede que satisfaz as restrições de conservação de fluxo e de satisfação da demanda. Para determinamos uma solução viável para o problema *PR* a partir de ϕ, podemos determinar uma ponderação inteira ω' (que está associada a um protocolo) dos conjuntos independentes de G_I satisfazendo (4.1). Isso equivale a encontrar uma coloração fracionária ponderada pelo fluxo ϕ. Portanto, a heurística *WFCP* pode ser aplicada ao grafo G_I para gerar a ponderação ω'. Basta apenas utilizar a função de peso p definida por $p(e) = \phi[\vec{e}] + \phi[\overleftarrow{e}]$, $\forall e \in E$. Como consequência, o fluxo por período de qualquer escalonamento do protocolo resultante é um múltiplo de ϕ.

Na Seção 4.4.2, descrevemos com detalhes uma heurística que desenvolvemos para determinar um fluxo inteiro inicial que satisfaz uma vez a demanda dos nós de origem a partir de multiplicadores lagrangeanos da relaxação lagrangeana do problema *PR*, definida na Seção 4.4. Além disso, mostramos uma adaptação da heurística *WFCP* que faz uso dos multiplicadores lagrangeanos para obter melhores resultados na geração de uma solução viável para o problema *PR*.

4.4 Relaxação Lagrangeana para a Formulação por Representantes do Problema PR

Em nossa pesquisa, trabalhamos com a relaxação lagrangeana da formulação por representantes descrita na Seção 4.2 com a adição das restrições $x^e[e] \leq 1$, $\forall e \in E$. De maneira similar à mostrada na Seção 3.2, para tornar independentes os subproblemas definidos para cada vértice e de G_I no grafo induzido por sua anti-vizinhança positiva, $\bar{G_I}^+(e)$,

as restrições relaxadas são as de (4.3). Observe que tais restrições acoplam variáveis de diferentes representantes, e, ao removê-las, conseguimos separar o problema em um subproblema para cada representante.

A relaxação do problema para um vetor de multiplicadores λ, $\lambda \geq \mathbf{0}$, fixo se torna:

$$\min \quad \sum_{e\in E} x^e[e] + \sum_{e\in E} \lambda[e](\phi[\vec{e}] + \phi[\overleftarrow{e}] - \sum_{e'\in \bar{N}^-[e]} x^{e'}[e])$$

$$\begin{array}{lcll} \text{s.a.} & x^e \in STAB_{x^e[e]}(\bar{G_I}^+(e)) & & e \in E \\ & 0 \leq x^e[e] \leq 1 & & e \in E \\ & \sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = 0 & & u \in V - (V_O \cup V_D) \\ & \sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = d(u) & & u \in V_O \end{array}$$

Reorganizando a função objetivo, chegamos ao problema $P(\lambda)$, definido como:

$$\min \quad \sum_{e\in E}(1-\lambda[e])x^e[e] + \lambda[e](\phi[\vec{e}] + \phi[\overleftarrow{e}]) - \sum_{e'\in \bar{N}^+(e)} \lambda[e']x^e[e']$$

$$\begin{array}{lcll} \text{s.a.} & x^e \in STAB_{x^e[e]}(\bar{G_I}^+(e)) & & e \in E \\ & 0 \leq x^e[e] \leq 1 & & e \in E \\ & \sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = 0 & & u \in V - (V_O \cup V_D) \\ & \sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = d(u) & & u \in V_O \end{array}$$

Como relaxamos as restrições que acoplavam as variáveis x e as variáveis de fluxo ϕ, podemos decompor o problema $P(\lambda)$ em dois subproblemas:

Subproblema de Coloração $P^c(\lambda)$:

$$\begin{array}{lcll} P^c(\lambda) & \min & \sum_{e\in E}(1-\lambda[e])x^e[e] - \sum_{e'\in \bar{N}^+(e)} \lambda[e']x^e[e'] & (4.8) \\ & \text{s.a.} & x^e \in STAB_{x^e[e]}(\bar{G_I}^+(e)) \quad e \in E & \\ & & 0 \leq x^e[e] \leq 1 \quad e \in E & (4.9) \end{array}$$

Subproblema de Fluxo $P^f(\lambda)$:

$$\begin{array}{lcll} P^f(\lambda) & \min & \sum_{e\in E} \lambda[e](\phi[\vec{e}] + \phi[\overleftarrow{e}]) & (4.10) \\ & \text{s.a.} & \sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = 0 \quad u \in V - (V_O \cup V_D) & \\ & & \sum_{v\in N(u)} \phi[u\vec{v}] - \phi[v\vec{u}] = d(u) \quad u \in V_O & \end{array}$$

Além disso, o subproblema P^c pode ser decomposto em subproblemas de conjunto independente máximo ponderado para cada $e \in E$. Analisando a função objetivo (4.8), observamos que, para cada $e \in E$, a restrição (4.9) pode ser substituída por $x^e[e] \in \{0, 1\}$, pois

$x^e[e] = 1$ resulta em um valor da função objetivo melhor ou igual a $x^e[e] = \epsilon$, $0 < \epsilon < 1$ (similar ao apresentado na Seção 3.2).

Considere o subproblema de *conjunto independente máximo ponderado* associado a e, para cada $e \in E$:

$$P_e^c(\lambda) \quad \max \sum_{e' \in \bar{N}^+(e)} \lambda[e']\hat{x}^e[e']$$

$$\text{s.a.} \quad \hat{x}^e \in STAB({\bar{G}_I}^+(e)) \tag{4.11}$$

Para $e \in E$, sejam $\sigma_e(\lambda)$ o valor ótimo do subproblema $P_e^c(\lambda)$ e $Rep(\lambda) = \{e \in E \mid \sigma_e(\lambda) > 1 - \lambda[e]\}$. $Rep(\lambda)$ representa o conjunto de vértices e de G_I que são representantes em uma solução ótima, ou seja, $x^e[e] = 1$ resulta em um valor da função objetivo de (4.8) melhor ou igual ao valor obtido fixando $x^e[e] = 0$.

Para encontrarmos a solução ótima x de $P^c(\lambda)$, resolvemos os subproblemas $P_e^c(\lambda)$, $e \in E$, separadamente, e, de acordo com a solução ótima $\hat{x}$, atribuimos os valores de x da seguinte forma:

$$x^e[e'] = \begin{cases} 1, & \text{se } e \in Rep(\lambda), e' = e \\ \hat{x}^e[e'], & \text{se } e \in Rep(\lambda), e' \neq e \\ 0, & \text{caso contrário} \end{cases} \qquad e \in E, e' \in \bar{N}^+[e]$$

Quanto ao subproblema $P^f(\lambda)$, observe que se trata de um problema de fluxo mínimo no Grafo da Rede sem restrição de capacidade nas arestas. Assim, podemos resolvê-lo com o seguinte algoritmo combinatório: para cada $u \in V_O$, encontramos um caminho mínimo ponderado orientado de u a um nó de destino t no Grafo da Rede, onde o peso de cada arco $vw \in A$ é igual ao valor do multiplicador lagrangeano da restrição relaxada (4.3) correspondente ($\lambda[vw] = \lambda[wv]$). O nó de destino t é tal que o caminho mínimo determinado até ele é o menor caminho dentre todos os caminhos de u a um nó de destino na rede. Em seguida, adicionamos um fluxo nesse caminho de valor igual a demanda do vértice de origem u. Dessa forma, temos um fluxo que satisfaz a demanda e minimiza (4.10). Em nossa implementação, utilizamos o algoritmo de Dijkstra (DIJKSTRA, 1959) para encontrar os caminhos mínimos.

Assim sendo, se $\sigma^f(\lambda)$ é o valor ótimo do subproblema de fluxo $P^f(\lambda)$, então o valor ótimo de $P(\lambda)$ é igual a:

$$\sigma^f(\lambda) + \textstyle\sum_{e \in Rep(\lambda)}(1 - \lambda[e] - \sigma_e(\lambda))$$

4.4.1 Método de Duas Fases no Problema PR

Assim como feito para o problema *CF*, dada a relaxação lagrangeana apresentada na seção anterior, para aplicarmos o *Método do Subgradiente de Duas Fases* ao problema *LD* no caso do problema *PR*, precisamos ainda definir quais heurísticas aplicar e como calcular o subgradiente.

O cálculo do subgradiente $\Delta(\lambda^k)$ na iteração k em relação ao vetor de multiplicadores λ^k é dado pela expressão:

$$\Delta(\lambda^k)[e] = \bar{\phi}_k[\vec{e}] + \bar{\phi}_k[\overleftarrow{e}] - \sum_{e' \in \bar{N}^-[e]} \bar{x}_k^{e'}[e], \qquad e \in E$$

onde $(\bar{x}_k, \bar{\phi}_k)$ é a solução ótima de $P(\lambda^k)$.

4.4.1.1 Fase 1

Na primeira execução da fase 1, usamos o limite superior UB fixo de valor igual ao valor da função objetivo (4.2) aplicada à solução retornada pela heurística *MinPathsWFCPRelLag* (definida na seção seguinte) com valor de parâmetro dos multiplicadores lagrangeanos iguais a 1 (heurística gulosa).

4.4.1.2 Fase 2

Na segunda fase, o limite superior é calculado em cada iteração do subgradiente também através do valor da função objetivo (4.2) aplicada à solução retornada pela heurística lagrangeana *MinPathsWFCPRelLag*. Desta vez, os valores dos multiplicadores usados são os valores dos multiplicadores corrente. A heurística faz uso dos multiplicadores lagrangeanos para determinar uma coloração fracionária ponderada pelo fluxo que ela mesma determina.

4.4.2 Heurística Lagrangeana MinPathsWFCPRelLag

A heurística lagrangeana que desenvolvemos para o problema *PR*, baseada na relaxação lagrangeana desta seção, é chamada *MinPathsWFCPRelLag*. Essa heurística é composta por duas heurísticas: *MinPathsRelLag* e *WFCPRelLag*. A heurística *MinPathsRelLag* determina um fluxo inteiro ϕ no Grafo da Rede que satisfaz as restrições de conservação de fluxo e de satisfação da demanda (demanda original de valor inteiro). A segunda heurística determina uma coloração fracionária ponderada pela função $p(e) = \phi[\vec{e}] + \phi[\overleftarrow{e}]$, $\forall e \in E$, obtendo um protocolo tal que qualquer escalonamento do mesmo produz, para cada orientação de aresta $\vec{e}, e \in E$, um fluxo múltiplo de $\phi[\vec{e}]$ no período correspondente. Ambas fazem uso dos multiplicadores lagrangeanos para que combinadas determinem uma boa solução.

Algoritmo 10: Algoritmo para obtenção de um fluxo inicial para o problema *PR*.

Chamada: MinPathsRelLag($G_{rede} = (V, E)$, d, λ)
Entrada: Grafo G_{rede}, demanda d e vetor de multiplicadores lagrangeanos λ.
Resultado: Fluxo em G_{rede} que satisfaz a demanda.

```
1  para cada e = uv ∈ E faça
2      w(u, v) ← w(v, u) ← λ[e]
3      φ[e⃗] ← φ[e⃖] ← 0
4  para cada u ∈ V_O faça
5      repita
6          cam ← Dijkstra(G_rede, u, t, w)
7          para cada e = uv ∈ cam faça
8              φ[e⃗] ← φ[e⃗] + 1
9              w(u, v) ← w(u, v) + λ[e]
10             w(v, u) ← w(v, u) + λ[e]
11         d[u] ← d[u] − 1
12     até que d[u] = 0
13 Reorganizar fluxo
14 retorne φ
```

O Algoritmo 10 mostra como a heurística *MinPathsRelLag* funciona. Para cada $u \in V_O$, é determinado um caminho mínimo orientado de u a um nó de destino t no Grafo da Rede, onde o peso $w(v, z)$ de cada arco $vz \in A$ é igual ao valor do multiplicador lagrangeano da restrição relaxada (4.3) correspondente ($\lambda[vw] = \lambda[wv]$). O nó de destino t é tal que o caminho mínimo determinado até ele é o menor caminho dentre todos os caminhos de u a um nó de destino na rede. Em seguida, é adicionado uma unidade de fluxo nesse caminho e, para cada arco vz desse caminho, ao seu peso e ao peso do arco zv é somado o valor $\lambda[vz] = \lambda[zv]$, que corresponde ao seu coeficiente no subproblema de fluxo. Dessa forma, tentamos gerar um fluxo ϕ que dispersa a demanda dos nós de origem mantendo o valor $\sum_{e \in E} \lambda[e](\phi[\vec{e}] + \phi[\overleftarrow{e}])$ próximo do valor ótimo $\sigma^f(\lambda)$ do subproblema de fluxo da relaxação. Isso é feito até que as restrições de satisfação de demanda sejam satisfeitas (ou seja, a demanda para todo vértice de origem seja atendida). Após esse procedimento, alteramos o fluxo de tal forma a não permitir, para cada aresta do Grafo da Rede, fluxo positivo nos dois sentidos. Isso é feito da seguinte maneira: para cada $uv \in E$ tal que $\phi[\vec{uv}] > 0$ e $\phi[\vec{vu}] > 0$, seja $\phi[\vec{vu}]$ o menor dentre esses dois valores, sem perda de generalidade. Para cada caminho direcionado p_1 que passa pelo arco vu até um vértice de destino d_1, seja p_2 um caminho que passa pelo arco uv até um vértice de destino d_2. Substituímos o subcaminho $v \to d_1$ pelo subcaminho $v \to d_2$ e o subcaminho $u \to d_2$ pelo subcaminho $u \to d_1$. Observe que as restrições de conservação de fluxo e de demanda ainda são satisfeitas. Após a realização de testes, verificamos que essa modificação do fluxo gera soluções viáveis melhores ao final do *MSDF*. Novamente, utilizamos o algoritmo de Dijkstra (DIJKSTRA, 1959) para a determinação dos caminhos mínimos.

Algoritmo 11: Algoritmo para obtenção de uma coloração inteira ponderada.

Chamada: WSCHRelLag($G_I[U] = (U, I[U])$, r)
Entrada: Grafo $G_I[U]$ e peso residual r dos vértices do grafo.
Resultado: Coloração inteira ponderada por r.

```
1  C ← ∅
2  para todo e ∈ U faça
3      c_λ[e] = 1 − λ[e] − Σ_{e'∈N̄⁻(e)∩U} λ[e]
4  repita
5      Crie uma classe de cor S e coloque o máximo de vértices e ∈ U possível, em
       ordem decrescente de peso residual e não-decrescente de c_λ
6      C ← C ⊎ {S}
7      r_min ← min{r(e) | e ∈ S}
8      y_S ← r_min
9      para cada e ∈ S faça
10         r(e) ← r(e) − r_min
11         se r(e) = 0 então
12             U ← U \ {e}
13             para cada e' ∈ U faça
14                 se e' ∈ N̄⁺(e) então
15                     c_λ[e'] ← c_λ[e'] + λ[e']
16 até que U = ∅
17 retorne (C, y)
```

A heurística *WFCPRelLag* para coloração fracionária ponderada é a heurística *WFCP*, onde usamos para geração da coloração inteira ponderada em cada iteração a heurística *WSCHRelLag*, dada pelo Algoritmo 11. Na heurística *WSCHRelLag*, a ordem de escolha dos vértices possui um segundo critério além do peso residual. A ideia é similar ao que é feito na Seção 3.4, onde queremos determinar uma coloração fracionária ponderada com o valor não muito distante do valor ótimo da relaxação lagrangeana $P(\lambda)$. Cada vértice $e \in E$ possui uma *nota* $c_\lambda[e]$ determinado dinamicamente pela soma dos coeficientes das variáveis de vértices que podem representar e. A ordem imposta pelo segundo critério é então a ordem não-decrescente dessas notas. Assim sendo, na heurística *WFCPRelLag*, a ordem em que os vértices são examinados na linha 8 do Algoritmo 5 é determinada pelo critério da ordem não-crescente de peso residual e, caso haja empate, o critério da ordem não-decrescente de nota.

4.4.3 Pricing

Assim como no problema *CF*, realizamos a técnica de *pricing* para reduzir o tempo de execução do *MSDF*. O procedimento é feito da mesma forma que a descrita na Seção 3.5, porém, substituindo os termos $FC(\lambda)$ e $FC_u(\lambda)$ por $P^c(\lambda)$ e $P^c_e(\lambda)$, respectivamente, já

Algoritmo 12: Algoritmo para obtenção de uma solução para o problema *PR*.

Chamada: MinPathsWFCPRelLag(G_{rede}, G_I, d, λ)

Entrada: Grafo da Rede G_{rede}, Grafo de Interferências G_I, demanda inteira d dos nós de origem e vetor de multiplicadores lagrangeanos λ.

Resultado: Solução para o problema *PR*.

1 $\phi \leftarrow$ MinPathsRelLag(G_{rede}, d, λ)
2 $(\mathcal{C}, y) \leftarrow$ WFCPRelLag(G_I, ϕ, λ)
3 **retorne** $(\mathcal{C}, y, \phi)$

que o problema *PR* é modelado em função do Grafo de Interferências G_I.

5 Resultados Computacionais

A avaliação das heurísticas lagrangeanas para os problemas *CF* e *PR* foi realizada através de experimentos computacionais e análise dos seus resultados. Neste capítulo, apresentamos esses resultados e análises. Em particular, apresentamos uma comparação com as respectivas heurísticas gulosas (*FCP* no caso do problema *CF*, e *WFCP* no caso do problema *PR*). A avaliação da heurística lagrangeana para o problema *PR* é complementada com comparações com a heurística *SER* proposta em (VIEIRA et al., 2012) para o problema *EC* formulado na Introdução desta dissertação. As instâncias usadas para essas avaliações foram geradas usando um simulador de redes, no caso, o *ns2*, de uma forma detalhada no decorrer do capítulo. As avaliações e comparações são feitas com relação ao tempo de execução e qualidade da solução encontrada. O trabalho de experimentos computacionais foi realizado em parte na COPPE/UFRJ com a colaboração do professor Valmir Carneiro Barbosa e do pesquisador em pós-doutorado Fabio Rocha Jimenez Vieira.

5.1 Configuração do Ambiente Computacional

A linguagem de programação usada foi a linguagem *C++*. Utilizamos o pacote `CPLEX` versão 12.4 para resolução dos subproblemas de programação linear. Tais subproblemas são resolvidos paralelamente com o uso da biblioteca *pthreads* do *C++*, com 32 threads para o problema *CF* e 4 threads para o problema *PR*. O pacote `CPLEX` utiliza informações de subproblemas já resolvidos, como a base do sistema linear do problema, para facilitar a resolução dos demais subproblemas e, por isso, os resultados gerados são não-determinísticos, pois a ordem de resolução dos subproblemas em cada iteração do subgradiente pelas threads não é determinada. Por isso, geramos os resultados de 5 execuções para cada instância de cada problema.

Os experimentos foram executados em uma máquina com processador de quatro núcleos de 3.4GHz, 7.8GB de memória RAM e sistema operacional *Linux* de 64 bits.

5.2 Parâmetros das Heurísticas Lagrangeanas

Realizamos a implementação do *MSDF* (*Método do Subgradiente de Duas Fases*) para o problema de *Coloração Fracionária* e o problema de *Ponderação de Rodadas*. Os parâmetros do método usados na implementação para os dois problemas foram:

- Passo ψ na fase 1 iniciado com valor 2;

- $N_{parado} = 10$;
- $S_\psi = 0.001$ (Impomos um limite máximo de 200 iterações na fase 1);
- Passo ψ na fase 2 com valor fixo igual a 1 para o problema *CF* e 1.5 para o problema *PR*;
- $N_{f2} = 100$.

Esses valores de parâmetros foram encontrados empiricamente, após vários experimentos. O valor inicial de $\psi = 2$ foi sugerido como em (REEVES, 1993).

Para as heurísticas de coloração *FCPRelLag* (problema *CF*) e *WFCPRelLag* (problema *PR*), impomos um limite de $N_{lim} = 1000$ cores para cada solução.

Os subproblemas de programação linear ($FC_u(\lambda)$ e $P_e^c(\lambda)$) resolvidos na implementação consistem em problemas de *conjunto independente máximo ponderado* (*CIMP*), por isso podem demandar um tempo de execução muito alto. O número de restrições em (3.18) e em (4.11) é potencialmente exponencial, então usamos as restrições de arestas (3.5) e o pacote `CPLEX` para geração de cortes para assim conseguir um limite inferior para o valor ótimo da relaxação lagrangeana dos problemas, quando usado no *MSDF*. Os parâmetros de geração de cortes do `CPLEX` usados em nossa implementação foram:

- *CPX_PARAM_CLIQUES* = 1 (Geração de cortes de clique moderadamente)
- *CPX_PARAM_FRACCUTS* = 1 (Geração de cortes de Gomory moderadamente)
- *CPX_PARAM_ZEROHALFCUTS* = 1 (Geração de cortes 0-1/2 moderadamente)

Esses parâmetros foram determinados após a realização de vários experimentos, onde verificamos os cortes mais utilizados pelo `CPLEX` na resolução dos subproblemas de *CIMP*.

5.3 Experimentos com o Problema CF

5.3.1 Instâncias

As instâncias dos experimentos para o problema *CF* foram retiradas da parte de coloração de grafos da DIMACS, encontradas em http://mat.gsia.cmu.edu/COLOR/instances.html. Elas foram escolhidas por serem consideradas instâncias relativamente difíceis para determinação do número cromático fracionário (CAMPêLO; CAMPOS; CORRêA, 2009), (GVOZDENOVIC, 2008). Para os grafos aleatórios *gx.y*, onde x é o número de vértices e y é a chance em porcentagem de uma aresta não ser escolhida na geração do grafo, as tabelas apresentadas nesta seção mostram a média de todas as informações para 5 grafos aleatórios com essa configuração. A tabela 1 contém as características gerais das instâncias, mostradas abaixo:

Tabela 1 – Informações de Instâncias para o Problema de Coloração Fracionária

Grafo	$\|V\|$	$\|E\|$	Densidade	χ_F
2-Insertions.4	149	541	4%	2.32ˆ
4-FullIns.3	114	541	8%	6.17
5-FullIns.3	154	792	6%	7.14ˆ
DSJC125.9	125	6961	90%	42ˆ
DSJC250.9	250	27897	90%	71.9ˆ
g200.10	200	17907	9%	-
myciel4	23	71	28%	3.24
myciel5	47	236	21%	3.55
myciel6	95	755	17%	3.83
queen6.6	36	290	46%	7
queen8.8	64	728	36%	8.44
queen9.9	81	1056	32%	9

- *Grafo*: Grafo de instância G
- $|V|$: número de vértices de G
- $|E|$: número de arestas de G (não direcionadas)
- Densidade: densidade de G em porcentagem
- χ_F: número cromático fracionário. O símbolo ˆ indica um limite inferior para esse valor

5.3.2 Resultados e Análises

As Tabelas 2 e 3 mostram os resultados obtidos, cujas informações são:

- *Grafo*: Grafo de instância G
- χ_F: número cromático fracionário. O símbolo ˆ indica um limite inferior para esse valor
- *FCP*: limite superior encontrado pela heurística *FCP*
- Best *LB*: melhor limite inferior encontrado dentre as execuções do *MSDF* sem pricing
- Avg *UB*: valor médio do limite superior encontrado nas execuções do *MSDF*
- Best *UB*: melhor limite superior encontrado dentre as execuções do *MSDF*
- *Iter*: valor médio do número de iterações do *MSDF*
- *Redução*: redução da diferença entre *FCP* e χ_F (ou limite inferior conhecido na literatura, caso χ_F não seja conhecido) obtida por Best *UB* em porcentagem
- *T(s)*: valor médio do tempo de execução do *MSDF* em segundos.

Tabela 2 – MSDF sem Pricing para o Problema de Coloração Fracionária

Grafo	χ_F	*FCP*	Best *LB*	Avg *UB*	Best *UB*	*Iter*	*Redução*	*T(s)*
2-Insertions.4	2.32^	3.33 (10/3)	2.4396	2.82	2.75 (22/8)	3	65%	1455
4-FullIns.3	6.17	6.40 (32/5)	6.0899	6.17	6.17 (37/6)	2	100%	178
5-FullIns.3	7.14^	7.20 (36/5)	7.0817	7.20	7.20 (36/5)	2	0%	291
DSJC125.9	42^	49.29 (345/7)	42.7041	47.45	47.27 (520/11)	2	31%	23
DSJC250.9	71.9^	83.71 (586/7)	70.3311	82.17	81.85 (1064/13)	3	16%	169
g200.10	-	71.56 (514/7)	60.5607	69.21	68.98 (825/12)	3	23%	84
myciel4	3.24	3.60 (18/5)	3.2430	3.40	3.40 (17/5)	2	56%	3
myciel5	3.55	4.00 (40/10)	3.5400	3.83	3.83 (23/6)	3	38%	122
myciel6	3.83	4.80 (24/5)	3.5649	4.41	4.33 (26/6)	2	48%	1618
queen6.6	7	7.67 (23/3)	6.9997	7.26	7.00 (7003/1000)	2	100%	9
queen8.8	8.44	10.75 (43/4)	8.2985	9.70	9.60 (48/5)	2	50%	111
queen9.9	9	11.17 (67/6)	8.9946	10.78	10.70 (107/10)	2	22%	180
Média de Cores	-	(237/6)	-	-	(814/70)	-	-	-

Tabela 3 – MSDF com Pricing para o Problema de Coloração Fracionária

Grafo	χ_F	*FCP*	Best *LB*	Avg *UB*	Best *UB*	*Iter*	*Redução*	*T(s)*
2-Insertions.4	2.32^	3.33 (10/3)	2.4396	2.79	2.75 (11/4)	2	62%	339
4-FullIns.3	6.17	6.40 (32/5)	6.0899	6.17	6.17 (37/6)	2	100%	96
5-FullIns.3	7.14^	7.20 (36/5)	7.0817	7.19	7.17 (43/6)	2	50%	183
DSJC125.9	42^	49.29 (345/7)	42.7041	47.49	47.12 (754/16)	2	33%	21
DSJC250.9	71.9^	83.71 (586/7)	70.3311	82.08	81.87 (1228/15)	2	16%	117
g200.10	-	71.56 (514/7)	60.5607	69.23	69.09 (886/12)	3	22%	61
myciel4	3.24	3.60 (18/5)	3.2430	3.45	3.40 (17/5)	2	56%	3
myciel5	3.55	4.00 (40/10)	3.5400	3.85	3.80 (38/10)	3	44%	62
myciel6	3.83	4.80 (24/5)	3.5649	4.38	4.29 (30/7)	2	53%	717
queen6.6	7	7.67 (23/3)	6.9997	7.29	7.00 (7002/1000)	2	100%	7
queen8.8	8.44	10.75 (43/4)	8.2985	9.71	9.67 (58/6)	2	47%	99
queen9.9	9	11.17 (67/6)	8.9946	10.82	10.75 (86/8)	2	19%	126
Média de Cores	-	(237/6)	-	-	(860/71)	-	-	-

A Tabela 2 é referente aos resultados sem a utilização da técnica de pricing, enquanto a Tabela 3 é referente aos resultados com o uso de pricing.

Observamos que o valor da melhor solução obtida pelo *MSDF* (coluna *BestUB*) é sempre melhor comparado ao valor da solução retornada pela heurística *FCP* utilizando *DSATUR* (experimentalmente, os melhores resultados são obtidos quando utilizamos o *DSATUR* como heurística de coloração inteira) como coloração inteira (coluna *FCP*), com uma redução de *gap* muito boa. Porém, o método demanda muito mais tempo de execução, já que a heurística *FCP* executa em tempo insignificante. Esse tempo é relativamente alto se levarmos em consideração o tamanho das instâncias que é relativamente pequeno. Por exemplo, o grafo *DSJC250.9* é o grafo com maior número de vértices, 250. A questão do tempo é devido a resolução dos muitos subproblemas lineares ao longo do *MSDF*. A técnica de pricing consegue reduzir bastante o tempo de execução sem piorar a qualidade da solução. Para os grafos *2-Insertions.4*, *5-FullIns.3*, *DSJC125.9*, *DSJC250.9* e os grafos aleatórios *g200.10* não conhecemos o número cromático fracionário. A média dos limites superiores encontrados pelo *MSDF* para uma mesma instância nos mostra que o não-determinismo do algoritmo não afeta significantemente o valor da solução retornada. Para os grafos *4-FullIns.3* e *queen6.6* conseguimos obter uma solução com valor ótimo, enquanto para o grafo *queen9.9* conseguimos uma redução de *gap* de apenas 19% e uma solução com valor não tão próximo do número cromático fracionário quanto desejado. O número de

iterações do *MSDF* varia em torno de 2 ou 3 iterações, o que podemos concluir que o trabalho feito em cada iteração do método é custoso. Um fato bastante positivo do método é que, além da melhora do valor de solução viável, o valor do melhor limite inferior encontrado em tal método é bem próximo do valor ótimo.

5.4 Experimentos com o Problema PR

5.4.1 Geração de Instâncias

De maneira similar a (VIEIRA et al., 2012), foram geradas 50 grafos de redes ($G = (V, E)$) posicionando os $|V|$ nós dentro de um quadrado com lado de comprimento 1500. Para cada rede, o primeiro nó foi posicionado no meio do quadrado. Dado o raio R_G de comunicação entre os nós, e com isso a relação de vizinhança entre os nós (dois nós são vizinhos entre si se, e somente se, a distância euclidiana entre eles não é maior que R_G), os nós restantes foram posicionados aleatoriamente, um por vez. Para a definição da posição de um nó, foram dadas as seguintes restrições: o nó deveria ter pelo menos um vizinho, nenhum nó poderia ter mais que Δ_G vizinhos e não poderiam existir dois nós com distância entre eles menor que 25 unidades euclidianas de distância. O número máximo de tentativas de posicionar um nó satisfazendo as restrições mencionadas foi de 1000. Quando esse limite foi atingido, a rede que estava em construção foi eliminada e uma nova construção de rede foi iniciada. Os valores de $R_G = 170$ e $\Delta_G = 8$ foram determinados de tal forma a obter redes com densidades próximas às que são encontradas na prática. Esse método gera redes bem próximas de possíveis redes reais para os problemas. Para todos os valores de $|V| \in \{60, 80, 100, 120, 140, 160, 180, 200, 220, 240\}$ foram geradas 5 redes, resultando nas 50 redes usadas nos experimentos desta pesquisa. O limite superior de 240 nós na rede foi determinado de acordo com o tamanho de redes reais, que, geralmente, não possuem mais que esse número de nós.

Para a definição dos nós de origem e destino, utilizamos a posição x dos nós no quadrado da rede, cuja dimensão é 2 (x e y). Dados os nós da rede ordenados pela posição x, definimos os N_O primeiros nós como nós de origem e os últimos N_D nós como nós de destino. Por padrão, a demanda de cada nó de origem foi determinada com o valor 2 a fim de gerar instâncias para o problema PR que não fossem facéis de serem resolvidas.

O Grafo de Interferências $G_I = (E, I)$ para as instâncias dos problemas PR e EC foi gerado a partir do critério de distância 2 entre os links da rede. Inicialmente $I = \emptyset$. Se uv e $wz \in E$ possuíam vértices em comum, ou seja, $u = w$ ou $u = z$ ou $v = w$ ou $v = z$, então (uv, wz) era adicionado a I. Depois disso, para cada uv e $wz \in E$, se a distância mínima entre uv e wz em G_I fosse menor ou igual a 2, então (uv, wz) era adicionado a I ao final do procedimento de geração do Grafo de Interferências.

Tabela 4 – MSDF sem Pricing para o Problema de Ponderação de Rodadas. $N_O = N_D = |V|/6$

Grafo	$\|E\|$	$\|I\|$	*WFCP*	Best *LB*	Avg *UB*	Best *UB*	Dev *UB*	*Iter*	*Redução*	*T(s)*
g_{int}.100	278	5174	60.20 (615/10)	31.4162	54.63	53.34 (725/14)	7.63	2	22.22%	229
g_{int}.120	336	6214	61.28 (469/8)	32.2044	56.88	56.04 (694/12)	9.87	3	13.96%	346
g_{int}.140	406	7885	74.40 (619/8)	33.5796	67.89	65.72 (635/10)	15.65	3	18.32%	529
g_{int}.160	486	10180	61.83 (616/9)	28.1683	58.49	56.10 (598/10)	5.50	3	13.93%	762
g_{int}.180	560	11979	69.81 (426/6)	33.2314	68.95	68.40 (490/7)	11.81	3	2.64%	1088
g_{int}.200	642	14421	59.02 (543/9)	29.6393	58.88	58.82 (685/11)	2.35	3	0.62%	1709
g_{int}.220	733	17228	63.51 (687/10)	28.3062	62.86	62.25 (744/11)	3.54	3	3.26%	2025
g_{int}.240	825	20000	61.84 (692/11)	31.4828	61.84	61.84 (692/11)	1.79	3	0.00%	2729
g_{int}.60	152	2482	47.27 (653/15)	35.8387	46.46	46.14 (668/16)	14.21	2	9.00%	60
g_{int}.80	219	4062	55.35 (762/14)	32.1522	52.37	51.54 (893/17)	9.73	3	14.69%	127
Média	-	-	(608/10)	-	-	(682/11)	-	-	-	-

5.4.2 Resultados e Análises

Como todas as redes das instâncias dos experimentos foram grafos aleatórios seguindo a construção explicada anteriormente, os resultados mostrados são a média de todas as informações para 5 grafos aleatórios com a mesma configuração, definida pelo número de nós da rede, exceto pelo desvio padrão que é referente aos melhores limites superiores das instâncias de uma mesma configuração. As Tabelas 4 e 5 mostram os resultados obtidos para $N_O = N_D = |V|/6$, cujas informações são:

- *Grafo*: Grafo de Interferências G_I
- $|E|$: número de vértices de G_I
- $|I|$: número de arestas de G_I (não direcionadas)
- *WFCP*: limite superior encontrado pela heurística *MinPathsWFCPRelLag* com valor do multiplicadores lagrangeanos igual a 1 (heurística gulosa)
- Best *LB*: melhor limite inferior encontrado dentre as execuções do *MSDF* sem pricing
- Avg *UB*: valor médio do limite superior encontrado pelo *MSDF*
- Best *UB*: melhor limite superior encontrado dentre as execuções do *MSDF*
- Dev *UB*: desvio padrão, para as instâncias de mesma configuração, do melhor limite superior encontrado nas execuções do *MSDF*
- *Iter*: valor médio do número de iterações do *MSDF*
- *Redução*: redução da diferença, em porcentagem, entre o resultado da heurística gulosa e o melhor limite inferior encontrado (pois o valor ótimo não é conhecido) obtida pelo limite superior na execução do *MSDF*
- *T(s)*: valor médio do tempo de execução do *MSDF* em segundos.

Tabela 5 – MSDF com Pricing para o Problema de Ponderação de Rodadas. $N_O = N_D = |V|/6$

Grafo	$\|E\|$	$\|I\|$	*WFCP*	Best *LB*	Avg *UB*	Best *UB*	Dev *UB*	*Iter*	*Redução*	*T(s)*
g_{int}.100	278	5174	60.20 (615/10)	31.4162	54.88	53.63 (854/16)	7.22	3	21.74%	196
g_{int}.120	336	6214	61.28 (469/8)	32.2044	57.21	56.03 (512/9)	8.71	3	13.76%	248
g_{int}.140	406	7885	74.40 (619/8)	33.5796	68.08	64.17 (642/10)	15.79	3	24.70%	372
g_{int}.160	486	10180	61.83 (616/9)	28.1683	57.57	56.66 (781/13)	6.24	3	12.63%	523
g_{int}.180	560	11979	69.81 (426/6)	33.2314	68.56	67.61 (501/7)	11.04	2	4.52%	638
g_{int}.200	642	14421	59.02 (543/9)	29.6393	58.67	57.64 (634/11)	3.17	2	4.35%	972
g_{int}.220	733	17228	63.51 (687/10)	28.3062	62.73	62.00 (899/14)	3.68	2	3.95%	1325
g_{int}.240	825	20000	61.84 (692/11)	31.4828	61.84	61.84 (692/11)	1.79	2	0.00%	1665
g_{int}.60	152	2482	47.27 (653/15)	35.8387	46.60	46.46 (722/17)	14.16	3	6.56%	48
g_{int}.80	219	4062	55.35 (762/14)	32.1522	52.95	52.19 (689/13)	9.44	3	12.99%	113
Média	-	-	(608/10)	-	-	(692/12)	-	-	-	-

A Tabela 6 é referente às informações de fluxo, complementando a tabela 5.

- *Grafo*: Grafo da Rede $G_{rede} = (V, E)$
- $|V|$: número de vértices em G
- $|E|$: número de arestas em G (não direcionadas)
- *dist*: menor distância entre um vértice de origem e um vértice de destino em número de arestas
- *Demanda Total*: soma das demandas de todos os vértices de origem do Grafo da Rede
- *Min* Flow: menor valor de fluxo em aresta não nula do Grafo da Rede na melhor solução encontrada nas execuções do *MSDF*
- *Max* Flow: maior valor de fluxo em aresta do Grafo da Rede na solução cujo valor é Best *UB*
- Avg Flow: média dos valores de fluxo em arestas não nulas do Grafo da Rede na melhor solução encontrada nas execuções do *MSDF*
- nFlowNull: número de arestas do Grafo da Rede com fluxo nulo na melhor solução encontrada nas execuções do *MSDF*.

Os valores de parâmetro $N_O = N_D = |V|/6$ para esses experimentos, referentes ao número de nós de origem e de destino, respectivamente, foram definidos empiricamente depois de testes realizados para determinar instâncias relativamente difíceis para o problema (usando como referência o tempo de execução do *MSDF*), dada a rede.

Observamos que, para o problema *PR*, os subproblemas lineares eram, em sua grande maioria, resolvidos de maneira ótima pelo `CPLEX`, o que permite concluir que tais subproblemas são fáceis. Porém, a grande quantidade de subproblemas a ser resolvido ao longo do *MSDF* é a causa do tempo relativamente alto mostrado nos resultados.

Tabela 6 – Distribuição do Fluxo pela Rede. MSDF com Pricing para o Problema de Ponderação de Rodadas. $N_O = N_D = |V|/6$

Grafo	$\|V\|$	$\|E\|$	*dist*	*Demanda Total*	*Min* Flow	*Max* Flow	Avg Flow	nFlowNull
g_{rede}.100	100	278	14	32	1	13	2.48	121
g_{rede}.120	120	336	15	40	1	12	2.50	128
g_{rede}.140	140	406	16	46	1	15	2.83	173
g_{rede}.160	160	486	16	52	1	12	2.36	190
g_{rede}.180	180	560	17	60	1	12	2.41	187
g_{rede}.200	200	642	17	66	1	7	2.00	165
g_{rede}.220	220	733	17	72	1	13	2.15	247
g_{rede}.240	240	825	18	80	1	10	2.01	217
g_{rede}.60	60	152	12	20	1	10	2.25	70
g_{rede}.80	80	219	13	26	1	11	2.43	107

O valor da melhor solução obtida pelo *MSDF* (coluna *BestUB*) é sempre melhor comparado ao limite superior encontrado pela heurística *MinPathsWFCPRelLag* usando como parâmetro os valores dos multiplicadores iguais a 1 (heurística gulosa, coluna *WFCP*), exceto para as redes com 240 nós, e a redução do *gap* é boa. Porém, o método demanda muito mais tempo de execução, já que a heurística *MinPathsWFCPRelLag* executa em tempo insignificante. A utilização da técnica de pricing reduz bastante o tempo de execução mantendo a qualidade do valor da solução. Para algumas instâncias, obtemos uma redução de mais de 30%. Para as redes com 100 e 140 nós, obtemos uma redução do *gap* de quase 25%, enquanto que para as redes com 60, 180, 200 e 220 número de nós, obtemos uma redução inferior a 10%. Para as redes com 240 nós, não conseguimos melhora no valor da solução comparando com a heurística gulosa. Não conhecemos o valor ótimo para nenhuma instância, mas acreditamos que o limite inferior obtido pelo *MSDF* seja próximo, pois a relaxação lagrangeana do problema *PR* foi feita de tal forma a diferir da relaxação lagrangeana do problema *CF* (que obtém bons limites inferiores) apenas pela adição do subproblema de fluxo, que conseguimos resolver de maneira ótima. A diferença entre o melhor limite superior encontrado e o limite inferior junto com o tempo de execução do método deixa evidente a dificuldade do problema. O desvio padrão mostra a grande diferença entre os valores das soluções para instâncias diferentes, mesmo possuindo a mesma configuração. O número de iterações do método é em média 3.

A utilização da heurística para esse problema traz como vantagem uma solução com poucas cores se comparada com uma solução da formulação linear-inteira 4.2 – 4.7 (ver questão do valor k encontrado possuir muitos dígitos, Seção 2.5). Isso influencia na aplicação de problemas práticos de *PR*, pois uma solução com muitas cores (rodadas) pode se tornar inviável.

Analisando as informações do fluxo, podemos observar que a menor distância entre um vértice de origem e um vértice de destino em número de arestas é suficientemente grande em relação à rede, comprovando uma boa dispersão entre os nós de origem e destino nas instâncias usadas nos experimentos. Pelo baixo valor da média dos fluxos não nulos em relação ao valor do fluxo máximo que passa em uma aresta do Grafo da Rede, podemos

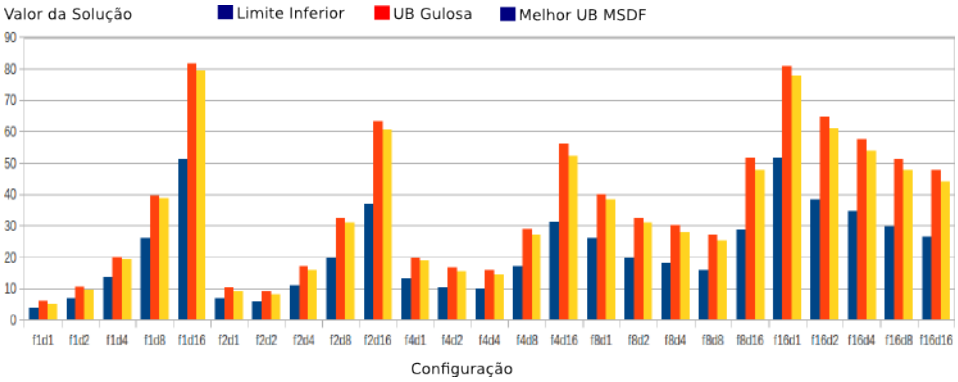

Figura 16 – Gráfico com os valores das soluções. A coluna azul (primeira) indica o valor do limite inferior encontrado pelo *MSDF*. A coluna vermelha (segunda) indica o valor da solução retornada pela heurística gulosa. A coluna amarela (terceira) indica o valor da melhor solução encontrada pelo *MSDF*. Os parâmetros N_O e N_D são mostrados na forma $f_i d_j$, onde $i = N_O$ e $j = N_D$.

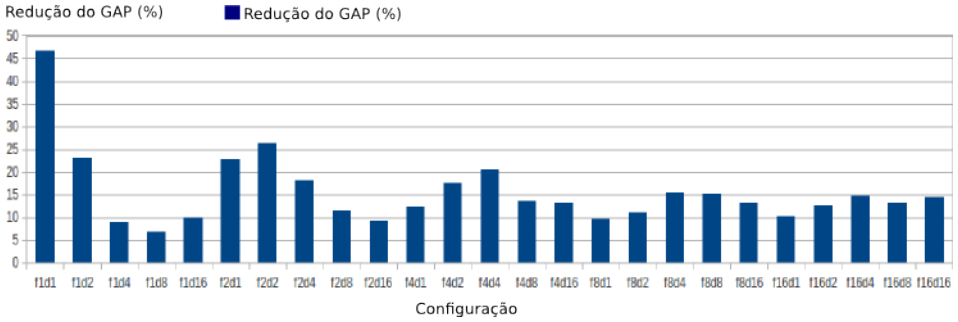

Figura 17 – Gráfico com a redução do *gap* (diferença entre o valor da solução da heurística gulosa e o limite inferior encontrado no *MSDF*) em porcentagem. Os parâmetros N_O e N_D são mostrados na forma $f_i d_j$, onde $i = N_O$ e $j = N_D$.

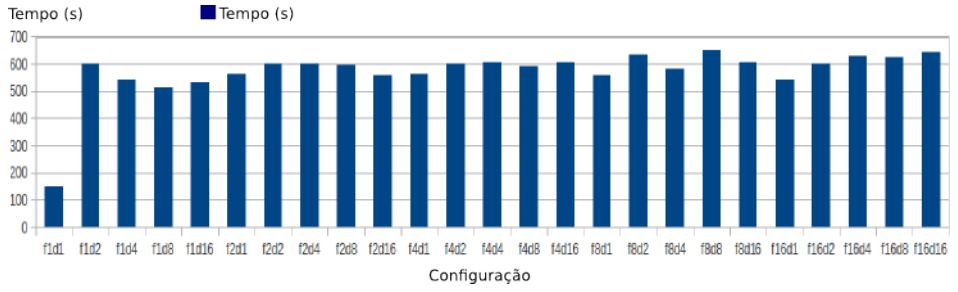

Figura 18 – Gráfico com a média do tempo de execução do *MSDF* em segundos. Os parâmetros N_O e N_D são mostrados na forma $f_i d_j$, onde $i = N_O$ e $j = N_D$.

concluir que o fluxo da melhor solução encontrada pelo *MSDF* dispersa adequadamente a demanda dos nós de origem a fim de tentar fazer o máximo uso do paralelismo de transmissões. O número de arestas do Grafo da Rede com fluxo nulo gira em torno de $|E|/3$ e $|E|/2$, o que corresponde a um bom uso da rede pela solução.

Para uma análise melhor da nossa heurística, realizamos também experimentos com a variação dos parâmetros de número de origens e destinos, N_O e N_D, para as redes geradas. Para toda combinação de valores N_O, $N_D \in \{1, 2, 4, 8, 16\}$, executamos o *MSDF* para todas as instâncias (lembramos que para cada instância o método é executado 5 vezes), calculamos as médias das informações para cada configuração de rede (instâncias cujo Grafo da Rede possuem o mesmo número de vértices) e, em seguida, calculamos a média de todos esses valores, convergindo para um único valor para cada informação (valor da solução, tempo de execução, ...). Para esses experimentos, a demanda dos nós de origem é definida como:

$$d(o) = \begin{cases} 2, & \text{se } N_O \geq N_D \\ \lceil (2 \times N_D)/N_O \rceil, & \text{caso contrário} \end{cases} \qquad o \in V_O$$

Os resultados são mostrados nos Gráficos 16, 17 e 18. Os parâmetros N_O e N_D são mostrados na forma $f_i d_j$, onde $i = N_O$ e $j = N_D$. No Gráfico 16, podemos perceber que os valores das soluções são maiores quando a diferença entre o número de origens e o número de destinos é maior. Isso é devido à concentração de transmissões ao redor dos nós de origem (se $N_O < N_D$) ou dos nós de destino (se $N_D < N_O$), minimizando a vazão total. Nesses mesmos casos, também podemos concluir, pelo Gráfico 17, que a redução do *gap* (diferença entre o valor da solução da heurística gulosa e o limite inferior encontrado no *MSDF*) é menor. Em particular, para o caso $N_O = N_D = 1$, temos uma grande redução do *gap*. Pelo Gráfico 18, podemos ver que a média do tempo de execução do *MSDF* é aproximadamente igual para todas as configurações dos parâmetros N_O e N_D, exceto a configuração $N_O = N_D = 1$, onde temos um tempo bem inferior, o que indica ser uma configuração de instância relativamente fácil de ser resolvida.

5.4.3 Comparação com o Algoritmo SER

O problema *EC* (formulado na Seção 1.5) permite modelar situações em que os nós possuem capacidade limitada de memória. No entanto, para que essa modelagem seja efetiva, é preciso estabelecer de antemão os caminhos que podem ser usados. Esse fato pode levar a uma redução de eficiência da rede, caso os caminhos estabelecidos levem a escalonamentos com vazão insatisfatória. Uma forma de estabelecer caminhos que levem em conta a capacidade de vazão intrínseca à rede G submetida às restrições de interferência é determiná-los a partir de uma solução do problema *PR*. Dada uma instância do problema *PR* e um fluxo $\phi : A \to \mathbb{N}$ que satisfaz a demanda dos nós de origem uma única vez

obtido pela heurística *MinPathsRelLag*, podemos particionar o fluxo em caminhos origem-destino, não necessariamente disjuntos ou diferentes, que escoam uma unidade de fluxo. Tais caminhos formam uma coleção de caminhos origem-destino que podem ser usados como entrada para o problema *EC*. Uma outra possibilidade é usar o fluxo definido em todo o período de um escalonamento viável para a instância do problema *PR*, que, neste caso, pode satisfazer a demanda dos nós de origem mais de uma vez.

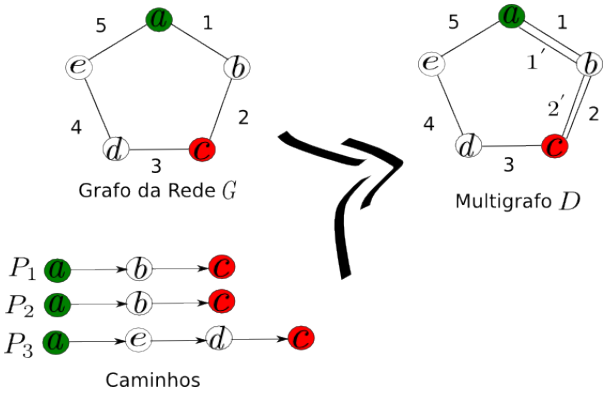

Figura 19 – Exemplo de geração do multigrafo D a partir de uma rede modelado por G e uma coleção de caminhos. O Grafo da Rede é um C_5 com arestas numeradas de 1 a 5 e vértices nomeados de a a e. O multigrafo D tem o mesmo conjunto de vértices de G e suas arestas são numeradas de 1 a 5, $1^{'}$ e $2^{'}$.

Como forma de analisar a qualidade de nossa heurística lagrangeana para o problema *PR*, realizamos uma comparação entre nossa heurística e o algoritmo *SER* (*Scheduling by Edge Reversal*), uma heurística para o problema *EC* (VIEIRA et al., 2012).

Considere a notação dada na Seção 1.5 para uma instância do problema *EC*. Seja $D = (X, Y)$ o multigrafo induzido pela coleção de caminhos. Denotamos por $G_I^{'} = (Y, I^{'})$ o grafo que representa explicitamente as interferências para as transmissões da coleção de caminhos dada como entrada. Nesse caso, cada transmissão de cada caminho corresponde a um vértice de $G_I^{'}$ e o conjunto de arestas $I^{'}$ indica as transmissões que se interferem quando executadas simultaneamente. Para $\vec{e}$ e $\vec{e'} \in Y$, sejam e e e' as arestas correspondentes a $\vec{e}$ e $\vec{e'}$ em G, respectivamente, não necessariamente distintas. Então, $(\vec{e}, \vec{e'}) \in I^{'}$ se, e somente se, $(e, e') \in I$. Ilustrações para a geração do multigrafo D e do grafo $G_I^{'}$ são mostradas nas Figuras 19 e 20, respectivamente.

Uma solução para o problema *EC* é um escalonamento, que não necessariamente corresponde a uma coloração fracionária no grafo $G_I^{'}$. Isso é, é possível que todas as soluções viáveis com valor ótimo para uma instância do problema *EC* não correspondam a uma coloração fracionária em $G_I^{'}$, o que é facilmente verificado. Para fins de comparação, o problema *EC* é resolvido heuristicamente pelo algoritmo *SER* (VIEIRA et al., 2012), que retorna uma coloração fracionária em $G_I^{'}$. Porém, a coloração fracionária deve satisfazer a restrição de limite de armazenamento igual a 1 por caminho para todo nó, o que restringe a colorações intercaladas: se e, e' são vértices vizinhos em $G_I^{'}$ e $c_1^e < c_2^e < ... < c_k^e$,

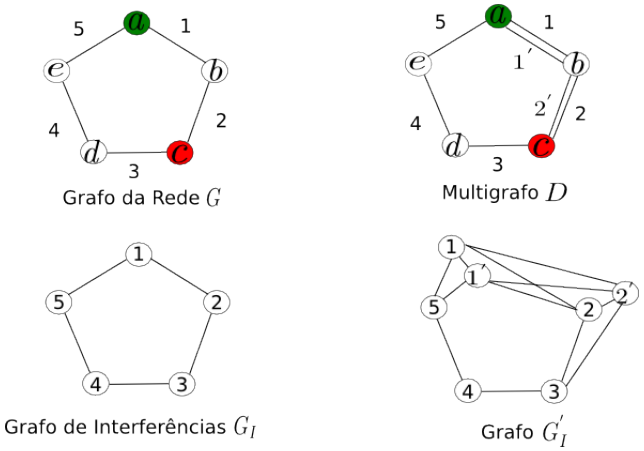

Figura 20 – Exemplo de geração do grafo G'_I a partir de uma rede modelado por G e seu Grafo de Interferências G_I. O Grafo da Rede é um C_5 com arestas numeradas de 1 a 5 e vértices nomeados de a a e. O multigrafo D tem o mesmo conjunto de vértices de G e suas arestas são numeradas de 1 a 5 e $1^{'}$ a $2^{'}$.

$c_1^{e'} < c_2^{e'} < ... < c_k^{e'}$ são as cores de e e e' respectivamente em uma coloração fracionária intercalada, então $c_1^e < c_1^{e'} < c_2^e < c_2^{e'} < ... < c_k^e < c_k^{e'}$ ou $c_1^{e'} < c_1^e < c_2^{e'} < c_2^e < ... < c_k^{e'} < c_k^e$.

A heurística definida pelo algoritmo *SER* tem sua ideia baseada na propriedade de que uma orientação acíclica no Grafo de Interferências gera vértices cujas orientações das arestas incidentes são todas direcionadas para si. Tais vértices são chamados de sinks. Dessa forma, o conjunto formado pelos vértices sinks de uma orientação acíclica representa um conjunto independente ou uma rodada. O algoritmo se inicia com uma orientação acíclica e a cada iteração inverte a orientação de todas as arestas incidentes aos vértices sinks, gerando uma nova orientação acíclica deterministicamente. Ele termina quando alcança uma orientação repetida, o que ocorre em algum momento, dado o número finito de orientações acíclicas no Grafo de Interferências. O multiconjunto formado pelas rodadas definidas pelas orientações acíclicas tem a propriedade de que cada vértice de G'_I aparece como vértice sink no mesmo número de orientações, e, além disso, dois vértices adjacentes aparecem como vértices sinks de maneira alternada (BARBOSA, 1996), (BARBOSA; GAFNI, 1989), caracterizando uma coloração fracionária alternada. A heurística então depende fortemente da orientação inicial, que pode ser definida como uma enumeração dos vértices de G'_I, onde a aresta entre dois vértices tem a orientação do maior para o menor. Basta então enumerar as arestas dos caminhos origem-destino. Em nossas implementações, usamos as quatro configurações iniciais descritas em (VIEIRA et al., 2012), *ND-BF*, *ND-DF*, *NI-BF*, *NI-DF* e mais duas configurações que consistem em ordenar os caminhos na ordem não-decrescente (ou ordem não-crescente) de número de arestas, numerar as arestas dos caminhos com percurso em profundidade a partir das origens dos caminhos seguindo a ordem definida pelo passo anterior, e reverter a numeração das arestas dos caminhos ímpares (Ex: o caminho que passa pelas arestas 7,1,3,4,9 invertido é correspondente ao caminho 9,4,3,1,7). Dentre as seis configurações iniciais, a que gera a

solução com melhor valor é a escolhida.

Pelo fato do algoritmo *SER* retornar uma solução que corresponde a uma coloração fracionária em $G_I^{'}$, usamos esse algoritmo no problema *EC* para a comparação com o *MSDF* para o problema *PR*. A rede e o Grafo de Interferências G_I usados nos dois problemas foram os mesmos. Para cada execução do *MSDF* para um instância do problema *PR*, os caminhos escoando fluxo entre origem e destino definidos pelo fluxo na rede determinado pela melhor solução viável encontrada (fluxo que satisfaz uma vez a demanda dos nós de origem, retornado pela heurística *MinPathsRelLag*) foram separados em caminhos de uma unidade de fluxo e formaram a coleção de caminhos origem-destino usada na instância de comparação no problema *EC*. O grafo $G_I^{'}$ correspondia então ao Grafo de Interferências G_I já com a ponderação do fluxo (para cada $e \in E$ e orientação $\vec{e}$, $\phi[\vec{e}]$ vértices em $G_I^{'}$ associados a $\vec{e}$). Sendo assim, uma coloração fracionária em $G_I^{'}$ era equivalente a uma coloração fracionária ponderada pelo fluxo em G_I. Dessa forma, conseguimos obter instâncias similares para os problemas *PR* e *EC*, e assim realizamos uma comparação de resultados.

As comparações com o algoritmo *SER* são mostradas nas Tabelas 7 e 8, cujas colunas são:

- *Grafo*: Grafo de Interferências G_I
- $|E|$: número de vértices de G_I
- $|I|$: número de arestas de G_I (não direcionadas)
- Best *LB*: melhor limite inferior encontrado dentre as execuções do *MSDF*
- Avg *UBser*: valor médio, para as instâncias de mesma configuração, do melhor limite superior encontrado nas execuções do *SER*
- Best *UBser*: melhor limite superior encontrado dentre as execuções do *SER*
- Dev *UBser*: desvio padrão, para as instâncias de mesma configuração, do melhor limite superior encontrado nas execuções do *SER*
- *Tser(s)*: valor médio do tempo de execução do *SER* em segundos.
- Avg *UB*: valor médio do limite superior encontrado pelo *MSDF*
- Best *UB*: melhor limite superior encontrado dentre as execuções do *MSDF*
- Dev *UB*: desvio padrão, para as instâncias de mesma configuração, do melhor limite superior encontrado nas execuções do *MSDF*
- *T(s)*: valor médio do tempo de execução do *MSDF* em segundos.

Tabela 7 – MSDF para o problema *PR* com $N_O = N_D = |V|/6$ e SER para o problema *EC*: Fluxo da Heurística *MinPathsRelLag*

Grafo	$\|E\|$	$\|I\|$	Best *LB*	Avg *UBser*	Best *UBser*	Dev *UBser*	*Tser(s)*	Avg *UB*	Best *UB*	Dev *UB*	*T(s)*
g_{int}.100	278	5174	32.5190	106.96	98.80 (98/1)	15.32	0	54.88	53.63 (854/16)	7.22	196
g_{int}.120	336	6214	34.0271	115.72	106.20 (106/1)	18.53	0	57.21	56.03 (512/9)	8.71	248
g_{int}.140	406	7885	35.6380	146.96	135.60 (135/1)	31.95	0	68.08	64.17 (642/10)	15.79	372
g_{int}.160	486	10180	29.1550	129.42	124.60 (124/1)	16.86	0	57.57	56.66 (781/13)	6.24	523
g_{int}.180	560	11979	34.3504	163.12	146.80 (146/1)	18.24	0	68.56	67.61 (501/7)	11.04	638
g_{int}.200	642	14421	30.7643	149.00	138.40 (138/1)	10.74	0	58.67	57.64 (634/11)	3.17	972
g_{int}.220	733	17228	28.8249	156.68	144.00 (144/1)	6.29	0	62.73	62.00 (899/14)	3.68	1325
g_{int}.240	825	20000	32.9369	165.40	158.20 (158/1)	5.91	0	61.84	61.84 (692/11)	1.79	1665
g_{int}.60	152	2482	36.6517	73.76	70.00 (70/1)	23.55	0	46.60	46.46 (722/17)	14.16	48
g_{int}.80	219	4062	32.8906	93.28	86.20 (86/1)	10.50	0	52.95	52.19 (689/13)	9.44	113
Média	-	-	-	-	120.88 (120/1)	-	0	-	-	-	-

Tabela 8 – Resultados MSDF (Ponderação de Rodadas com $N_O = N_D = |V|/6$) e SER: Fluxo por Período

Grafo	$\|E\|$	$\|I\|$	Best *LB*	Avg *UBser*	Best *UBser*	Dev *UBser*	*Tser(s)*	Avg *UB*	Best *UB*	Dev *UB*	*T(s)*
g_{int}.100	278	5174	31.4162	114.08	104.12 (1099/11)	14.82	24	54.88	53.63 (854/16)	7.22	196
g_{int}.120	336	6214	32.2044	122.32	116.20 (1342/12)	16.68	33	57.21	56.03 (512/9)	8.71	248
g_{int}.140	406	7885	33.5796	149.34	139.96 (1223/9)	37.85	65	68.08	64.17 (642/10)	15.79	372
g_{int}.160	486	10180	28.1683	136.24	128.06 (1452/10)	16.60	113	57.57	56.66 (781/13)	6.24	523
g_{int}.180	560	11979	33.2314	176.73	167.08 (1146/6)	18.87	69	68.56	67.61 (501/7)	11.04	638
g_{int}.200	642	14421	29.6393	156.30	147.15 (1348/9)	7.43	169	58.67	57.64 (634/11)	3.17	972
g_{int}.220	733	17228	28.3062	166.33	157.08 (1528/9)	10.10	229	62.73	62.00 (899/14)	3.68	1325
g_{int}.240	825	20000	31.4828	176.29	166.19 (1870/11)	5.63	301	61.84	61.84 (692/11)	1.79	1665
g_{int}.60	152	2482	35.8387	79.18	75.02 (1041/15)	25.31	10	46.60	46.46 (722/17)	14.16	48
g_{int}.80	219	4062	32.1522	95.32	89.22 (1583/18)	11.57	15	52.95	52.19 (689/13)	9.44	113
Média	-	-	-	-	129.01 (1363/11)	-	102	-	-	-	-

A diferença do valor da melhor solução obtida pelo *MSDF* (coluna *BestUB*) e o limite superior encontrado pelo algoritmo *SER* (coluna Best *UBser*) é grande. Porém, novamente, o método demanda muito mais tempo de execução, já que o algoritmo *SER* executa em tempo insignificante. Notamos que as soluções retornadas pelo *SER* são sempre colorações inteiras para as instâncias deste experimento, o que explica a baixa qualidade da solução. Isso se deve à restrição de coloração alternada.

A Tabela 8 é referente aos resultados do algoritmo *SER* quando usado o fluxo referente a todo o período da melhor solução encontrada pelo *MSDF* para definir a coleção de caminhos usado na instância do problema *EC* correspondente. Dessa forma, a solução encontrada pelo *MSDF* satisfaz a restrição de memória de retenção de dados de tamanho unitário, assim como a solução retornada pelo *SER*. Porém, essa abordagem nos leva a resultados piores do algoritmo *SER*, como visto na tabela. Além do valor da solução ser pior em relação à outra abordagem de geração da coleção de caminhos, o tempo nesse caso não é mais insignificante. As colorações retornadas pelo *SER* ainda são inteiras. O valor da solução mostrado na tabela é fracionário pelo fato do fluxo por período (usado para definir a coleção de caminhos) satisfazer mais de uma vez a demanda dos nós de origem da instância do problema *PR*.

6 Conclusão

6.1 Resultados Obtidos e Contribuições

Desenvolvemos e implementamos uma heurística lagrangeana para o problema *CF* (problema de *Coloração Fracionária*). Tal heurística foi usada no *MSDF* (*Método do Subgradiente de Duas Fases*) para determinação de uma solução aproximada para o problema. A heurística é baseada em uma relaxação lagrangeana de uma formulação por representantes para o problema *CF* que possui duas características interessantes. A primeira delas é que o problema relaxado pode ser decomposto em subproblemas independentes e, consequentemente, podem ser resolvidos em paralelo. A segunda característica é o fato de conter restrições do poliedro de conjuntos independentes. Isso nos permite usar técnicas já conhecidas de separação de várias de suas facetas. Essas duas características foram exploradas na nossa estratégia de resolução utilizando o *MSDF*. Realizamos uma análise dos resultados fazendo uma comparação com a heurística *FCP*. Obtivemos melhores resultados em detrimento do tempo de execução, conseguindo uma boa redução do *gap* entre o valor ótimo (ou melhor limite inferior conhecido, nos casos em que o valor ótimo era desconhecido) e o valor da solução encontrada pela heurística *FCP*. Além da obtenção de melhores soluções viáveis, obtemos limites inferiores próximos do número cromático fracionário. O tempo de execução do *MSDF* é relativamente alto se analisarmos-o como uma heurística. Porém, ao contrário dos principais métodos de resolução do problema, o *MSDF* nos dá uma solução aproximada para o problema e não só o valor dessa solução.

O mesmo método de resolução usado para o problema *CF* foi adaptado para o problema *PR* (problema de *Ponderação de Rodadas*). No caso da relaxação lagrangeana do problema *PR*, além dos mesmos subproblemas de conjunto independente máximo ponderado da relaxação do problema *CF*, um novo subproblema surgiu. Porém, tratava-se de um problema de fluxo resolvido por um algoritmo combinatório polinomial. Realizamos uma análise dos resultados fazendo uma comparação com uma heurística gulosa. Obtivemos melhores resultados em detrimento do tempo de execução, conseguindo uma boa redução do *gap* entre o melhor limite inferior encontrado pelo *MSDF* e o valor da solução encontrada pela heurística gulosa. Como as instâncias usadas nos experimentos foram todas instâncias aleatórias, o valor ótimo era desconhecido. Nesse sentido, não podemos definir precisamente a qualidade do melhor limite inferior encontrado pelo *MSDF*. Porém, se analisarmos em relação ao problema *CF*, onde o *MSDF* encontra bons limites inferiores, a única diferença de resolução do problema relaxado é a adição da resolução de um subproblema de fluxo que conseguimos resolver de maneira ótima. Entretanto, a redução do *gap* para o problema *PR* é inferior à redução para o problema *CF*.

A aplicação prática do problema *PR* demanda não somente o valor da solução ótima, mas também a solução em si, especificando todas as rodadas. Por isso, é preferível o uso do *MSDF* ao invés do uso de um método como o de geração de colunas. Além disso, a heurística lagrangeana utilizada no *MSDF* gera uma solução com poucas rodadas se comparada com uma solução de uma formulação de programação linear-inteira para o problema *PR*. Isso melhora a viabilidade do uso da solução na aplicação prática.

Se considerarmos a aplicação que consiste em otimizar a vazão e o limite de tamanho da memória de armazenamento de dados de cada nó da rede, percebemos que a abordagem de resolução utilizando o problema *EC* (problema de *Escalonamento de Caminhos*) difere da abordagem utilizando o problema *PR*. No primeiro caso, o problema de fluxo é resolvido separadamente, visto que o problema *EC* trata do problema de coloração junto com o problema de otimização da memória. No caso do uso do problema *PR*, o problema de fluxo é resolvido junto com o problema de coloração. Portanto, há mais flexibilidade para otimizar a vazão na rede. Nessa abordagem, para otimizar o limite de tamanho da memória de armazenamento de dados de cada nó da rede, resolve-se o problema *ER* (problema de *Escalonamento de Rodadas*) em seguida, passando como entrada a solução do problema *PR*.

6.2 Trabalhos Futuros

Como trabalhos futuros, pensamos no desenvolvimento de heurísticas para os subproblemas de conjunto independente máximo ponderado (*CIMP*) com o intuito de acelerar o *MSDF* para os problemas *CF* e *PR*.

Uma análise do uso de informações de subproblemas resolvidos em iterações anteriores no *MSDF* para as iterações seguintes é também interessante, pois apenas as funções objetivo dos subproblemas *CIMP* são alteradas.

Na abordagem utilizada para resolução dos problemas apresentada neste trabalho, podemos observar vários pontos que ainda podem ser melhorados. Um exemplo é o método utilizado para encontrar multiplicadores lagrangeanos com bom limite dual. Existem outros métodos, além do método do subgradiente, que podem ser experimentados, como *Método de Bundle*, *Método de Dilatação de Espaço* e *Método de Projeção de Subgradiente*. Além disso, para a heurística lagrangeana, pode haver melhores configurações de parâmetros que geram melhores soluções. Como a quantidade de opções de configuração é grande, não garantimos a melhor configuração possível para a heurística lagrangeana, apesar dos resultados satisfatórios.

Referências

BALAS, E.; XUE, J. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica*, v. 15, n. 5, p. 397–412, 1996.

BARBOSA, V. C. *An Introduction to Distributed Algorithms*. Cambridge, MA: The MIT Press, 1996.

BARBOSA, V. C.; GAFNI, E. Concurrency in heavily loaded neighborhood-constrained systems. *ACM Trans. Program. Lang. Syst.*, ACM, New York, NY, USA, v. 11, n. 4, p. 562–584, out. 1989. ISSN 0164-0925. Disponível em: ⟨http://doi.acm.org/10.1145/69558.69560⟩.

BREGALDA, P. F.; BORNSTEIN, C. T.; OLIVEIRA, A. A. F. de. *Introdução à Programação Linear*. [S.l.]: Campus, 1981.

BRÉLAZ, D. New methods to color the vertices of a graph. *Commun. ACM*, ACM, New York, NY, USA, v. 22, n. 4, p. 251–256, abr. 1979. ISSN 0001-0782. Disponível em: ⟨http://doi.acm.org/10.1145/359094.359101⟩.

CAMPêLO, M.; CAMPOS, V. A.; CORRêA, R. C. Um algoritmo de planos-de-corte para o número cromático fracionário de um grafo. *Pesquisa Operacional*, scielo, v. 29, p. 179–193, 2009. ISSN 0101-7438. Disponível em: ⟨http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-74382009000100009&nrm=iso⟩.

CAPRARA, A.; FISCHETTI, M.; TOTH, P. A heuristic method for the set covering problem. *Operations Research*, v. 47, p. 730–743, 1995.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik*, Springer-Verlag, v. 1, n. 1, p. 269–271, 1959. ISSN 0029-599X. Disponível em: ⟨http://dx.doi.org/10.1007/BF01386390⟩.

FARIAS, P. M. S. *O Problema de Ordenação de Rodadas e Problemas de Otimização Associados*. Tese (Doutorado) — Universidade Federal do Ceará (UFC), Fortaleza - Ceará, Brazil, January 2014.

FISHER, M. L. The lagrangian relaxation method for solving integer programming problems. *Management Science*, v. 27, n. 1, p. 1–18, 1981. Disponível em: ⟨http://www.jstor.org/stable/30046157⟩.

GOMES, C. *Radio Mesh Networks and the Round Weighting Problem*. Tese (Doutorado) — Université de Nice-Sophia Antipolis (UNS), Sophia Antipolis, France, December 2009.

GUALANDI, S.; MALUCELLI, F. Exact solution of graph coloring problems via constraint programming and column generation. *INFORMS Journal on Computing*, v. 24, n. 1, p. 81–100, 2012.

GVOZDENOVIC, N. *Approximating the stability number and the chromatic number of a graph via semidefinite programming*. Tese (Doutorado) — Faculty of Science, April 2008.

HANSEN, P.; LABBÉ, M.; SCHINDL, D. Set covering and packing formulations of graph coloring: Algorithms and first polyhedral results. *Discrete Optimization*, v. 6, n. 2, p. 135 – 147, 2009. Also as Tec. Rep. G-2005-76 Cahier du GERAD, Montreal. Disponível em: ⟨http://www.optimization-online.org/DB_HTML/2005/12/1257.html⟩.

HELD, S.; COOK, W.; SEWELL, E. C. Maximum-weight stable sets and safe lower bounds for graph coloring. *Math. Program. Comput.*, v. 4, n. 4, p. 363–381, 2012.

KLASING, R.; MORALES, N.; PéRENNES, S. On the complexity of bandwidth allocation in radio networks. *Theoretical Computer Science*, v. 406, n. 3, p. 225 – 239, 2008. ISSN 0304-3975. Disponível em: ⟨http://www.sciencedirect.com/science/article/B6V1G-4SWFNWM-6/2/e4ff181ddfed1374882e17cdc3bbd220⟩.

LUND, C.; YANNAKAKIS, M. On the hardness of approximating minimization problems. *J. ACM*, ACM, New York, NY, USA, v. 41, n. 5, p. 960–981, 1994. ISSN 0004-5411. Disponível em: ⟨http://doi.acm.org/10.1145/185675.306789⟩.

MALAGUTI, E.; MONACI, M.; TOTH, P. An exact approach for the vertex coloring problem. *Discrete Optimization*, Elsevier, v. 8, n. 2, p. 174–190, 2011.

MEHROTRA, A.; TRICK, M. A. A Column Generation Approach for Graph Coloring. *Informs Journal on Computing*, v. 8, p. 344–354, 1996.

REEVES, C. R. *Modern Heuristic Techniques for Combinatorial Problems*. New York: Halsted Press, 1993.

VIEIRA, F. R. et al. Scheduling links for heavy traffic on interfering routes in wireless mesh networks. *Computer Networks*, v. 56, n. 5, p. 1584 – 1598, 2012. ISSN 1389-1286. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1389128612000357⟩.