ALGORITMO RELAX-AND-CUT PARA O PROBLEMA DO CONJUNTO INDEPENDENTE MÁXIMO

Matheus Caminha Pereira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Abilio Pereira de Lucena Filho

Rio de Janeiro
Setembro de 2018

ALGORITMO RELAX-AND-CUT PARA O PROBLEMA DO CONJUNTO INDEPENDENTE MÁXIMO

Matheus Caminha Pereira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Abilio Pereira de Lucena Filho, Ph.D.

Prof. Luidi Gelabert Simonetti, D.Sc.

Prof. Simone de Lima Martins, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2018

Pereira, Matheus Caminha

Algoritmo Relax-and-Cut para o Problema do Conjunto Independente Máximo/Matheus Caminha Pereira. – Rio de Janeiro: UFRJ/COPPE, 2018.

XI, 44 p.: il.; 29, 7cm.

Orientador: Abilio Pereira de Lucena Filho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 40 – 44.

1. Conjunto Independente Máximo. 2. Algorítimo Non Delayed Relax-and-Cut. 3. Algoritmo Exato. I. Lucena Filho, Abilio Pereira de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

"Creio muito na sorte.
Quanto mais trabalho, mais sorte
pareço ter." −Coleman Cox

Agradecimentos

Antes de todos, gostaria de agradecer a Dra. Marcia por tudo. Sempre me incentivou a aprender e a dar valor à vida acadêmica. Espero sempre responder à altura de todo amor, carinho e dedicação que recebi de você. Todas as minhas conquistas, são também suas!

Ao meu irmão, André, saiba que estarei sempre com você, não importa a distância. Vibro com todas as suas conquistas e sei que você também vibra com as minhas.

A Joana, minha companheira, viver contigo tem sido um privilégio. Poder contar com você, principalmente nos momentos mais difíceis, é a melhor sensação do mundo. Obrigado.

Aos amigos da COPPE, agradeço por toda ajuda, principalmente no início do mestrado. Vim de estatística e aprender computação foi muito mais fácil com o suporte de todos vocês. Em especial, ao Lucas, um amigo que levarei para vida toda. Muito obrigado por toda ajuda nesses anos, seja com estudos, códigos ou simplesmente as conversas no Labotim que tivemos até nos sábados! Saiba que você pode sempre contar comigo, irmão!

Ao meu orientador Abílio. Só tenho a agradecer por ter participado dessa jornada comigo. Sempre esteve disposto a ajudar, aconselhar e dar lições valiosas que levarei comigo sempre. Agradeço pela confiança e por toda paciência, que por muitas vezes foi testada, comigo. Não poderia pensar em um orientador melhor. Aprendi muito e ainda tenho muito a aprender com você. Meu mais sincero obrigado!

Por fim, tenho que agradecer aos meus chefes Marcelo e Nat, que sempre me apoiaram nesse projeto. A confiança de vocês em mim é um grande motivador para continuar estudando e me aperfeiçoando cada vez mais para me tornar um profissional ainda melhor.

Obrigado a todos vocês.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ALGORITMO RELAX-AND-CUT PARA O PROBLEMA DO CONJUNTO INDEPENDENTE MÁXIMO

Matheus Caminha Pereira

Setembro/2018

Orientador: Abilio Pereira de Lucena Filho

Programa: Engenharia de Sistemas e Computação

Propomos algoritmos exatos e heurísticos para o Problema do Conjunto Independente Máximo. Ao invés da formulação padrão do problema, que normalmente leva a limites de relaxação linear muito fracos, utilizamos uma reformulação muito mais forte, baseada numa cobertura de arestas por cliques. A reformulação é original e é aqui introduzida. Para resolvê-la, desenvolvemos um algoritmo do tipo Non Delayed Relax-and-Cut, um análogo Lagrangeano de um algoritmo de planos de corte. O algoritmo gera limites primais e duais para o problema. Isso é feito dualizando desigualdades válidas dinamicamente, à medida em que são violadas por soluções de Subproblemas Lagrangeanos. A seguir, num procedimento de *warm start*, desigualdades de clique (dualizadas dinamicamente durante a aplicação do algoritmo Relax-and-Cut) são agregadas à reformulação, que é então submetida, nessa forma reforçada, ao software CPLEX. O algoritmo Relax-and-Cut, atuando isoladamente e restrito apenas à dualização dinâmica de desigualdades de clique, conseguiu obter limites primais ou duais ótimos para 107 das 119 instâncias testadas. Por sua vez, o algoritmo exato, Relax-and-Cut-CPLEX, conseguiu obter certificados de otimalidade para 64 instâncias, três delas previamente em aberto há trinta anos. Além disso, conseguiu obter limites primais ou duais ótimos para 110 instâncias.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

RELAX-AND-CUT ALGORITHM FOR THE MAXIMUM INDEPENDENT SET PROBLEM

Matheus Caminha Pereira

September/2018

Advisor: Abilio Pereira de Lucena Filho

Department: Systems Engineering and Computer Science

Heuristic and exact algorithms are proposed for the Maximum Independent Set Problem. Instead of the standard formulation of the problem, which usually leads to very weak linear relaxation bounds, we use a much stronger reformulation, based on an edge clique cover. The reformulation is new and is introduced here. To solve it, we developed a Non Delayed Relax-and-Cut algorithm, a Lagrangian analog to a cutting plane algorithm. The algorithm generates primal and dual bounds for the problem. It does that while dynamically dualizing valid inequalities, as they become violated by solutions to Lagrangian Subproblems. Next, in a warm start procedure, clique inequalities (that were dynamically dualized throughout the Relax-and-Cut algorithm) are appended to the reformulation, which is then submitted, in that reinforced form, to solver CPLEX. The Relax-and-Cut algorithm, just by itself and restricted to dualizing only clique inequalities, was able to attain optimal primal or dual bounds for 107 of the 119 instances tested. On the other hand, the exact Relax-and-Cut-CPLEX algorithm obtained optimality certificates for 64 instances, three of them previously open for thirty years. Additionally, it also attained optimal primal or dual bounds for 110 instances.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Seja $G = (V, E)$ um grafo não direcionado, definido por um conjunto de vértices V e um conjunto de arestas E. Um subconjunto $D \subseteq V$ é um Conjunto Independente (CI) se nenhuma aresta de G tem ambas as extremidades em D. Encontrar um CI com a maior cardinalidade possível é denominado Problema do Conjunto Independente Máximo (PCIM) [1–3] e é um problema NP-difícil, como demonstrado em [4]. O exemplo de um CI é mostrado na Figura 1.1.

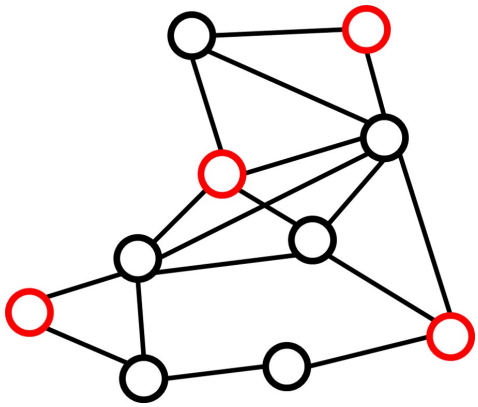

Figura 1.1: Exemplo de um Conjunto Independente Máximo (em vermelho)

O PCIM é equivalente ao Problema da Clique Máxima (PCM) [5–7] quando este é definido sobre o grafo complemento de G, $G' = (V, E')$ (veja [7], para maiores detalhes). Neste caso, $e = \{i, j\} \in E'$ se e somente se $e = \{i, j\} \notin E$. Note que uma clique de G' é um subconjunto de vértices $C \subseteq V$ tal que seu subgrafo induzido em G', ou seja, $G'[C]$, é completo. $G'[C]$ e por consequência G', têm então uma aresta ligando qualquer par de vértices distintos $i, j \in C$. Consequentemente, uma solução ótima $D \subset V$ para o PCIM (definido sobre G), é também ótima para o PCIM (definido sobre G').

Neste trabalho, introduzimos uma reformulação para o PCIM que se mostrou muito mais forte que a formulação padrão do problema. Baseado nessa reformulação,

desenvolvemos um algoritmo Non Delayed Relax-and-Cut (NDRC) [8] para resolver o PCIM. Como qualquer algoritmo NDRC, este também opera como um análogo Lagrangeano de um algoritmo de planos de corte. Além disso, como é usual para algoritmos NDRC, gera tanto limites duais (ou seja, limites superiores, nesse caso específico) quanto limites primais ou seja, soluções viáveis para o problema, nesse caso específico.

Quando por si só o algoritmo NDRC é incapaz de resolver à otimalidade uma dada instância do PCIM, passamos então para a fase seguinte de nosso procedimento de solução. Especificamente, utilizamos a informação gerada pelo algoritmo NDRC para inicializar, num estágio avançado, um algoritmo exato de enumeração implícita. No momento, tal algoritmo é aquele disponibilizado pelo software de Programação Inteira Mista (PIM) CPLEX [9]. No entanto, futuramente, pretendemos substituí-lo por um algoritmo Branch-and-Cut [10], especificamente projetado para resolver o PCIM.

Note que o procedimento descrito acima pode ser entendido tanto como uma heurística, em sua primeira fase, quanto como um algoritmo exato, combinadas as suas duas fases. Note também que nosso algoritmo NDRC, por gerar simultaneamente tanto limites primais quanto duais, é capaz de eventualmente oferecer certificados de otimalidade para as instâncias que se quer resolver. Isso ocorre quando os dois limites coincidem. Vale ainda ressaltar que essa capacidade de gerar simultaneamente limites primais e duais, comum a todas as heurísticas Lagrangeanas, não é oferecida pelas metaheurísticas propostas para o PCIM (vide [11–13], dentre outras).

Nesta dissertação, utilizando apenas o algoritmo NDRC, conseguimos obter certificados de otimalidade para 34 das 119 instâncias de teste que utilizamos. Sob as mesmas condições, conseguimos também obter limites duais ou limites primais iguais aos valores ótimos (ou iguais aos melhores limites primais conhecidos, quando as instâncias estão em aberto) para 107 instâncias. Aplicando o algoritmo completo, o número de certificados de otimalidade sobe para 64, três destes para instâncias que estavam em aberto há mais de 30 anos. Da mesma forma, o número de instâncias para as quais limites inferiores ou os limites superiores são iguais ao valores ótimos, sobe para 110.

1.1 Revisão Bibliográfica

Na literatura, algumas denominações alternativas são utilizadas para se referir ao PCIM. Por exemplo, ele é também chamado de Problema de Conjunto Estável Máximo [14] ou Problema de Empacotamento de Nós de Máxima Cardinalidade [15].

As três afirmações que se seguem sobre um subconjunto $S \subset V$ são equivalentes, intuitivas e conhecidas na literatura [5]: (a) S é um CI máximo de G, (b) S é uma clique máxima de $\bar{G}$, (c) $V \setminus S$ é uma cobertura mínima para os vértices de G. Dessa forma, os três problemas que se seguem são equivalentes: (a) encontrar o CI Máximo em G, (b) encontrar uma Clique Máxima de $\bar{G}$, (c) encontrar uma Cobertura Mínima para os Vértices de G. Todos este, como sabemos [16], são NP-Completos. Tais problemas têm diversas aplicações (veja BALAS e YU e (1986, para detalhes acerca das mesmas). Dentre estas podemos citar:

- *Recuperação da Informação.* Se os vértices de $G = (V, E)$ definem porções de informações armazenadas e suas arestas representarem a relação entre essas porções, o problema de recuperação de informações totalmente relacionadas é equivalente a encontrar uma clique máxima de G.

- *Concepção Experimental.* Assuma que os vértices de $G = (V, E)$ representam subconjuntos (blocos) de tamanho n de um dado conjunto S de tratamentos em um determinado experimento estatístico e que suas arestas correspondem a pares de subconjuntos que possuem, no máximo, k elementos em comum. Então, o problema de encontrar um conjunto máximo de blocos com tamanho k é equivalente ao problema de encontrar uma clique máxima de G.

- *Transmissão de Sinal.* Se os vértices de $G = (V, E)$ definem sinais e suas arestas indicam pares de sinais que são claramente distintos um do outro, o conjunto máximo de sinais claramente distinguíveis é, mais uma vez, equivalente ao problema de encontrar uma clique máxima de G.

Cada uma das aplicações mencionadas acima podem ser redefinidas como problemas de CI máximos (definidos sobre o grafo complemento de $G = (V, E)$).

Como indicado por REBENNACK *et al.* (2011), diversos algoritmos exatos baseados em técnicas de Programação Inteira já foram propostos para o PCIM. Dentre estes podemos citar: MANNINO e SASSANO (1994, 1996), BOMZE *et al.* (1999), BUTENKO e PARDALOS (2003), WARRIER *et al.* (2005), AVENALI (2007) e REBENNACK (2006) e CAMPELO e CORREA (2009). Em particular, o algoritmo proposto por CAMPELO e CORREA (2009) se baseia em Relaxação Lagrangeana.

Uma outra vertente de algoritmos exatos existentes para o PCIM é a dos Algoritmos Combinatórios. Dentre os propostos na literatura, podemos citar: TARJAN e TROJANOWSKI (1977), com tempo $2^{\frac{n}{3}} n^{O(1)}$ (foi o primeiro algoritmo não trivial proposto para resolver o PCIM), JIAN (1986), com tempo $1,2346^n n^{O(1)}$, ROBSON (2001), com tempo $1,2109^n n^{O(1)}$ e XIAO e NAGAMOCHI (2017) com tempo $1,1996^n n^{O(1)}$.

1.2 Organização da Dissertação

Organizamos essa dissertação em seis capítulos, incluindo este de introdução. No Capítulo 2, apresentamos a formulação padrão do PCIM e discutimos a reformulação que sugerimos para o problema. Esta reformulação, além de envolver um menor número de restrições, é muito mais forte que a formulação padrão. Isso pode ser comprovado, ainda no Capítulo 2, pelos resultados de Relaxação Linear (RL) obtidos para ambas. Estes foram obtidos para os seguintes conjuntos padrão de instâncias de teste para o PCIM: BROCK, C, C-FAT, DSJC, GEN, HAMMING, JHONSON, KELLER, MANN, P-HAT, SAN e FRB. A seguir, no Capítulo 3, apresentamos uma descrição genérica de algoritmos do tipo NDRC, seguido de uma especialização dos mesmos para o PCIM. Da mesma forma, discutimos como procedemos à identificação das desigualdades que dualizamos dinamicamente, como análogos Lagrangeanos de planos de corte (veja [28] para um algoritmo de planos de corte para o PCIM). No Capítulo 4, descrevemos inicialmente nossa heurística construtiva para a obtenção de uma solução viável para o PCIM. Feito isso, discutimos os procedimentos de busca local que utilizamos para melhorar a solução. No Capítulo 5, apresentamos os resultados computacionais obtidos com os nossos algoritmos. Finalmente, no Capítulo 6, fazemos algumas observações sobre nosso trabalho e oferecemos sugestões para trabalhos futuros.

Capítulo 2

Formulações para o Problema do Conjunto Independente Máximo

Nesta seção, descrevemos a formulação padrão do PCIM e a reformulação que utilizamos para o problema. Tal reformulação, como veremos, se baseia num recobrimento das arestas do grafo $G = (V, E)$, por cliques. Para obtê-lo, utilizamos uma heurística simples e que demanda pouco esforço computacional. Como indicado anteriormente, além de envolver um número significativamente menor de desigualdades, nossa reformulação também resulta em limites de RL muito mais fortes do que aqueles obtidos pela da formulação padrão.

2.1 Formulação Padrão

Associe uma variável $x_i \in \mathbb{R}$ à cada vértice $i \in V$ do grafo $G = (V, E)$ e considere uma região poliedral $\mathcal{R}_0$ definida por

$$x_i + x_j \leq 1, \{i, j\} \in E \tag{1}$$

$$x_i \geq 0, i \in V. \tag{2}$$

A formulação padrão do PCIM é descrita como

$$\text{Maximizar } \{\sum_{i \in V} x_i : x_i \in \mathcal{R}_0 \cap \mathbb{Z}^{|V|}\}, \tag{3}$$

e tem

$$\text{Maximizar } \{\sum_{i \in V} x_i : x_i \in \mathcal{R}_0\} \tag{4}$$

como sua relaxação linear.

Note que a formulação impõe, através de (1), que no máximo um vértice de cada

aresta de G pode fazer parte de um CI. Como característica marcante, a formulação padrão do PCIM tem uma Relaxação Linear (RL) que tende a ser muito fraca. Esse fato, como veremos adiante, é demonstrado pelos resultados computacionais que vamos apresentar.

Para cada conjunto de instâncias de teste, apresentamos na Tabela 2.1, (a) o número de instâncias envolvidas, **Total**, (b) o número médio de variáveis, **N**, (c) o número médio de restrições, **M**. Apresentamos também seus *gaps* médios de relaxação linear, **GAP**, expressos como os valores médios de 100.[(relaxação linear - valor ótimo) / (valor ótimo)], para cada instância de um dado conjunto (quando o valor ótimo é desconhecido, utilizamos seu melhor limite inferior conhecido), e (d) número de instâncias com o valor ótimo conhecidos, **N-OPT**.

Família	**Total**	**N**	**M**	**GAP**	**N-OPT**
BROCK	12	467	46.479	883	12
C	7	1.411	751.733	2.800	0
C-FAT	7	371	66.006	580	7
DSJC	2	750	155.900	2.528	0
GEN	5	320	5.584	236	5
HAMMING	6	448	18.171	430	6
JOHNSON	4	179	4.322	625	4
KELLER	3	1.436	368.797	1.588	2
MANN	4	1.195	2.309	48	3
P-HAT	13	692	170.883	1.783	6
SAN	15	347	30.562	919	15
FRB	41	1.086	81.828	985	41

Tabela 2.1: Gaps de dualidade para a formulação padrão.

Formulações mais fortes do que (3), são conhecidas para o PCIM. Por exemplo, aquela utilizada em [15], que envolve todas as cliques de G. Uma clique, como vimos anteriormente, é definida por um subconjunto de vértices $C \subset V$ que induz um subgrafo completo, $G[C]$, de G. Denotando por $\mathcal{C}$ o conjunto de todas as cliques distintas de G e utilizando as mesmas variáveis $\mathbf{x} \in \mathbb{R}^{|V|}$ definidas anteriormente, considere uma região poliedral $\mathcal{R}_1$ definida como

$$\sum_{i \in C} x_i \leq 1, \text{ para toda clique } C \in \mathcal{C} \tag{5}$$

$$x_i \geq 0, i \in V. \tag{6}$$

Uma formulação de PCIM [15] é então dada por

$$\text{Maximizar } \{\sum_{i \in V} x_i : x_i \in \mathcal{R}_1 \cap \mathbb{Z}^{|V|}\}, \tag{7}$$

e tem

$$\text{Maximizar } \{\sum_{i \in V} x_i : x_i \in \mathcal{R}_1\} \tag{8}$$

como sua relaxação linear.

A formulação (7) é pelo menos tão forte quanto (3). Na realidade, sua relaxação linear, (8), é normalmente muito melhor do que (4), ou seja, assume um valor muito menor que aquela. No entanto, quando G é um grafo geral, até mesmo a obtenção de (8) é um problema NP-difícil (vide [29], por exemplo). A situação é remediada ao se restringir as restrições (5) a um subconjunto $\mathcal{C}'$ de $\mathcal{C}$. A relaxação linear que resulta dessa *relaxação combinatória* de (7) fornece um limite superior para o valor de (8). Melhor ainda, este limite, ao contrário do anterior, pode ser obtido de forma eficiente.

Neste trabalho, optamos por uma alternativa intermediária entre utilizar a formulação (3) e a formulação (7). Seguindo o espírito da relaxação combinatória sugerida acima, vamos garantir, no entanto, que o subconjunto $\mathcal{C}' \subset \mathcal{C}$ que iremos escolher define uma reformulação de PCIM e não apenas uma relaxação do problema. Para tanto, nos basta garantir que $\mathcal{C}' \subset \mathcal{C}$ corresponde a uma cobertura por cliques das arestas de G.

2.2 Cobertura de Arestas por Cliques

O problema de recobrimento das arestas de um grafo geral por um número mínimo de cliques é NP-difícil (veja [30]). No entanto, a heurística que utilizamos para resolvê-lo é extremamente rápida e nos leva a reformulações do PCIM com limites de RL significativamente mais fortes que aqueles definidos por (4). Tal heurística funciona de forma gulosa, identificando cliques de G sucessivamente, até o ponto em que uma cobertura das arestas de G é obtida. À cada iteração, tentamos encontrar as maiores cliques ainda disponíveis.

Tanto quanto sabemos, a reformulação do PCIM que aqui propomos, embora extremamente simples e direta, não parece ter sido sugerida anteriormente. Como veremos, nossos algoritmos para o problema, tanto o algoritmo NDRC quanto o exato, se baseiam nessa reformulação.

2.3 Uma Heurística Gulosa para Recobrimento de Arestas por Cliques

A heurística que utilizamos opera de forma gulosa, foi sugerida por [31, 32], e implementada em [33]. Ela atua identificando cliques de $G = (V, E)$, que são então adicionadas a um conjunto que irá eventualmente definir uma cobertura por cliques das arestas de G. Quando o vértice não forma uma clique com nenhuma das cliques já existentes, uma nova clique é inicializada com aquele vértice.

Para descrever a heurística, definimos (a) a vizinhança aberta de $i \in V$, $\Gamma(i) = \{j \in V : \{i, j\} \in E\}$, (b) o conjunto de vizinhos de i com índice menor que i, $\Gamma^*(i) = \{j \in \Gamma(i) : j < i\}$, (c) um conjunto $\mathcal{K}$ com m cliques de cobertura, sendo $m = 0$, inicialmente, e (d) $n = |V|$.

O algoritmo percorre o conjunto de vértices V em ordem crescente de seus elementos. Assumindo que estamos na iteração i, onde $1 \leq i \leq n$, o vértice $i \in V$ é então aquele que está sendo investigado. Se $\Gamma^*(i) = \emptyset$, o que necessariamente se aplica pelo menos a $i = 1$, inicialize uma nova clique, fazendo $m := m+1$, $C_m = \{i\}$, e $C_m \in \mathcal{K}$. Caso contrário, investigue as cliques de $\mathcal{K}$ em ordem crescente de seus índices, de 1 a m, enquanto $|\Gamma^*(i)| \neq \emptyset$ ocorre. Para cada $C_k \in \mathcal{K}$, $1 \leq k \leq m$, verifique se $C_k \cup \{i\}$ é uma clique. Se for, tome $C_k := C_k \cup \{i\}$ e $\Gamma^*(i) := \Gamma^*(i) \setminus (\Gamma^*(i) \cap C_k)$. Ao longo deste procedimento, se $|\Gamma^*(i)|$ cair a zero, volte ao início do ciclo principal fazendo $i := i + 1$. Caso contrário, se $|\Gamma^*(i)| > 0$ ao final da iteração, um novo processo de geração de cliques é iniciado. Isso é feito concentrando-se apenas nos vértices de $\Gamma^*(i)$. Para tanto, identifique $C_{max} \in \Gamma^*(i)$, onde C_{max} é a maior clique existente em $\Gamma^*(i)$, e faça $C_{max} := C_{max} \cup \{i\}$. A seguir, inicie uma nova clique em $\mathcal{K}$, fazendo $m := m + 1$, $C_m = c_{max}$ e $\Gamma^*(i) := \Gamma^*(i) \setminus (\Gamma^*(i) \cap C_m)$. Repita esse procedimento até que $\Gamma^*(i)$ se torne vazio.

Na sequência, apresentamos no Algoritmo 1 um pseudo-código para o procedimento descrito acima.

Algorithm 1 Heurística para o Cobrimento Mínimo de Arestas por Cliques

Require: $G = (V, E)$ and a list of cliques $\mathcal{K} := \emptyset$

Ensure: a clique edge cover of G given by $\mathcal{K}$

```
n ← |V|
m ← 0
for (i = 1, ..., n) do
  Γ*(i) := {j ∈ Γ(i) : j < i}
  if |Γ*(i)| == 0 then
    m ← m + 1
    C_m ← {i}
    K ← K ∪ {C_m}
  else
    for (j = 1, ..., m) do
      if (C_j ∪ {i} is a clique) then
        C_j ← C_j ∪ {i}
        Γ*(i) ← Γ*(i) \ (C_j ∩ Γ*(i))
        if (|Γ*(i)| == 0) then stop
      end if
    end for
    while |Γ*(i)| > 0 do
      c_max ← c_k {where c_k is the maximum clique ∈ Γ*(i)}
      m ← m + 1
      C_m ← c_max ∪ {i}
      K ← K ∪ {C_m}
      Γ*(i) ← Γ*(i) \ C_m
    end while
  end if
end for
return K
```

Como saída do Algoritmo 1, obtemos um recobrimento das arestas de G por cliques. Não necessariamente um recobrimento contendo o menor número possível de cliques, mas um que, de qualquer modo, nos leva a uma reformulação do PCIM. Tal reformulação, como veremos a seguir, tende a ser muito mais forte do que a formulação padrão do problema e normalmente envolve um número muito menor de restrições.

2.3.1 Limites de Relaxação Linear para a Reformulação

Apresentamos na Tabela 2.2 informação similar àquela fornecida pela Tabela 2.1. Desta vez, no entanto, relativa à nossa reformulação do PCIM. $\mathbf{M'}$ indica o número médio de restrições da reformulação (por conjunto de instâncias de teste) e $\mathbf{GAP'}$ os gaps médios de dualidade por ela obtidos. Além disso, a Tabela 2.2 traz também as seguintes informações adicionais, que permitem uma comparação de nossa reformulação com a formulação padrão:

- **M (Redução):** percentual médio de redução no número de restrições;
- **GAP (Redução):** percentual médio de redução no valor da relaxação linear.

Família	Total	N	M′	GAP′	M (Redução)	GAP (Redução)
BROCK	12	467	7.072	206	19%	33%
C	7	1.411	52.542	512	33%	43%
C-FAT	7	371	8.259	2	17%	28%
DSJC	2	750	12.616	444	9%	21%
GEN	5	320	2.469	26	46%	41%
HAMMING	6	448	2.130	34	53%	59%
JOHNSON	4	179	31	0	5%	22%
KELLER	3	1.436	18.237	84	8%	15%
MANN	4	1.195	1.539	7	67%	72%
P-HAT	13	692	10.163	223	10%	25%
SAN	15	347	2.820	66	30%	26%
FRB	41	1.086	9.677	10	14%	11%

Tabela 2.2: Reformulação do PCIM: redução obtida no número de restrições e ganhos nos gaps de dualidade (em relação à formulação padrão).

Na média de todas as instâncias testadas, nossa reformulação contém 33% do número de restrições da formulação padrão. Da mesma forma, o limite superior obtido por sua relaxação linear corresponde a apenas 35% do valor da relaxação linear da formulação padrão. Como é possível notar, a vantagem comparativa de nossa reformulação por cobertura de arestas por cliques é expressiva. Tal vantagem, vale aqui ressaltar, foi obtida com um mínimo de esforço computacional. Além disso, é plausível imaginar que melhores coberturas de arestas por cliques, ou seja reformulações do PCIM que levem a relaxações lineares ainda mais fortes e envolvam

um número ainda menor de restrições, possam ser obtidas aplicando uma Busca Local à solução do Algoritmo 1.

Capítulo 3

Non Delayed Relax-and-Cut

Neste capítulo, descrevemos um algoritmo Non Delayed Relax-and-Cut (NDRC) genérico e a seguir uma especialização do mesmo para o PCIM. Apresentamos também um breve histórico sobre esses algoritmos e indicamos alguns problemas para os quais eles já foram propostos e testados.

Para o caso específico do PCIM, consideramos duas famílias de desigualdades válidas para reforçar nossa formulação do problema. Mais especificamente, trabalhamos com *desigualdades de cliques* e *desigualdades de buraco ímpar*.

Como veremos, as desigualdades de clique são as que mais contribuem para reforçar nossa formulação. Para simplificar a exposição, são também as que utilizamos para descrever o algoritmo NDRC. Concluída a descrição, discutimos algumas ideias investigadas com o intuito de tentar melhorar o desempenho do algoritmo. Feito isso, apresentamos, finalmente, o tratamento dispensado às desigualdades de ciclo ímpar. Estas, vale aqui ressaltar, não apresentaram um bom desempenho e as razões para tanto não estão claras. Talvez o tempo disponível para investigá-las tenha sido demasiadamente curto. De qualquer forma, pretendemos voltar a investigá-las, futuramente.

3.1 Algoritmos Non Delayed Relax-and-Cut

Algoritmos NDRC foram originalmente propostos por LUCENA (1992), sob um designação distinta da atual. Posteriormente, LUCENA (2005) sugeriu a denominação atualmente empregada para identificá-los. O objetivo sendo o de unificar, sob uma mesma nomenclatura, os diferentes algoritmos que operam como análogos Lagrangeanos de algoritmos de planos de cortes.

O termo Relax-and-Cut foi originalmente proposto por ESCUDERO *et al.* (1994) para denominar unicamente o algoritmo de ABOUDI *et al.* (1991). No entanto, na nomenclatura sugerida por LUCENA (2005), o algoritmo em [36] passou a se chamar *Delayed Relax-and-Cut* (DRC), ou seja um tipo particular de algoritmo Relax-and-

Cut. Uma diferença fundamental entre algoritmos NDRC e DRC é que o primeiro impõe, como veremos, uma dualização dinâmica de desigualdades à cada solução do Subproblema Lagrangeano (SL). O segundo, por outro lado, posterga esta dualização até que o Problema Dual Lagrangeano (PDL), como um todo, seja resolvido (veja LUCENA (2005), para maiores detalhes sobre os dois tipos de algoritmos Relax-and-Cut existentes). Pelas informações disponíveis [8], algoritmos NDRC apresentam um melhor desempenho que seus algoritmos DRC correspondentes.

Algoritmos NDRC têm sido utilizados com alguma frequência na literatura. Dentre vários outros, foram aplicados ao Problema de Steiner em Grafos por LUCENA(1992) e LUCENA(2005), ao Problema da Participação Retangular por CALHEIROS *et al.*(2003), ao Problema de Roteamento de Veículos por MARTINHON *et al.*(2004), ao Problema da Árvore Geradora com Restrição de Grau por DA CUNHA e LUCENA (2007), ao Problema da Partição de Conjuntos por CAVALCANTE *et al.*(2008), ao Problema da Árvore de Steiner com Recolha de Prêmios por DA CUNHA *et al.* (2009), ao Problema da Mochila 0-1 por DA CUNHA *et al.*(2010, e ao Problema do Subgrafo Conexo de Maior Peso por ÁLVAREZ-MIRANDA e SINNL (2017). Foram também utilizados na geração automática de desigualdades de Chvátal-Gomory para resolução de problemas gerais de Programação Inteira por FISCHETTI e SALVAGNIN (2011).

Todas as aplicações mencionadas acima foram implementadas utilizando essencialmente variantes do Método do Subgradiente (MS) (veja BEASLEY (1993), por exemplo, para maiores informações sobre o MS). A implementação de um algoritmo NDRC baseada no Método de Feixes foi proposta por BELLONI e SAGASTIZÁBAL (2009), que optaram no entanto, por excluir o termo Relax-and-Cut da denominação que deram ao algoritmo.

Como indicado anteriormente, algoritmos NDRC são análogos Lagrangeanos de algoritmos de planos de cortes e dualizam dois tipos básicos de restrições: (a) restrições que fazem parte da formulação do problema e (b) desigualdades válidas que não fazem parte da formulação, mas a tornam mais forte. Embora exceções existam, restrições do tipo (a) são normalmente dualizadas de uma forma estática, como é padrão nos algoritmos Lagrangeanos tradicionais. As restrições do tipo (b), no entanto, são dualizadas de forma dinâmica, à medida que violam a solução de um SL. Restrições do tipo (b) são descartadas se seus multiplicadores caem à zero no decorrer da aplicação do MS. Podem, no entanto, voltar a ser dualizadas se voltarem a violar a solução de um SL, no futuro.

Para descrever nosso algoritmo NDRC, considere a reformulação do PCIM que sugerimos no capítulo 2. Ou seja, aquela em que o conjunto de todas as cliques do grafo $G = (V, E)$, $\mathcal{C}$, é substituído por uma cobertura $\mathcal{C}'$ das arestas de G por cliques. Para facilitar a descrição do algoritmo, assumimos que algumas iterações do MS já

foram implementadas e que algumas cliques do tipo (b) (vide parágrafo anterior) já se encontram dualizadas. Assumimos então que a reformulação do PCIM disponível na iteração corrente do MS é:

$$\text{Maximizar} \sum_{i \in V} x_i \tag{9}$$

sujeito a

$$\sum_{i \in C} x_i \leq 1, \text{ para toda clique } C \in \mathcal{C}' \tag{10}$$

$$\sum_{i \in C} x_i \leq 1, \text{ para toda clique } C \in \mathcal{C}'' \tag{11}$$

$$x_i \in \{0, 1\}, i \in V, \tag{12}$$

onde $\mathcal{C}''$ corresponde às cliques dualizadas dinamicamente.

Associe multiplicadores de Lagrange $\lambda' \in \mathbb{R}_+^{|\mathcal{C}'|}$ e $\lambda'' \in \mathbb{R}_+^{|\mathcal{C}''|}$, respectivamente às desigualdades (10) e (11), obtendo assim o SL que denotamos por $\text{SL}(\lambda', \lambda'')$, ou seja:

$$\text{Maximizar} \sum_{i \in V} w_i x_i + \text{CT}(\lambda', \lambda'') \tag{13}$$

sujeito a

$$x_i \in \{0, 1\}, i \in V, \tag{14}$$

onde $\{w_i : i \in V\}$ são custos Lagrangeanos para as variáveis $\mathbf{x}$ e $\text{CT}(\lambda', \lambda'')$ é uma constante que se origina da dualização aplicada.

3.2 Implementação do Método do Subgradiente

Assuma que um limite inferior, l_{inf}, é conhecido para o PCIM e foi obtido, por exemplo, através da aplicação de uma heurística que gera soluções viáveis para o problema. Da mesma forma, assuma que um limite superior, l_{sup}, é também conhecido. Note, nesse caso, que $l_{sup} = |V|$ pode ser utilizado inicialmente. Tal limite, no entanto, é extremamente fraco e pode ser atualizado à cada solução de $\text{SL}(\lambda', \lambda'')$, pelo valor da solução ótima correspondente àquele SL. Isso, obviamente, quando tal valor implica numa redução do valor de l_{sup}.

Uma solução ótima para $\text{SL}(\lambda', \lambda'')$ é obtida fixando-se em 1 um subconjunto apropriado das variáveis $\mathbf{x}$, com as variáveis restantes sendo fixadas em 0. Se menos do que l_{inf} variáveis têm custos $\mathbf{w}$ positivos, as l_{inf} de maior custo, positivos ou negativos, são então fixadas em 1. Se, por outro lado, mais do que l_{sup} variáveis têm custos positivos, fixamos em 1 apenas as l_{sup} variáveis de maior custo. Finalmente, se um número de variáveis entre l_{inf} e l_{sup} têm custos $\mathbf{w}$ positivos, apenas estas são

fixadas em 1.

Numa dada iteração do MS, sob multiplicadores de Lagrange $\lambda = (\lambda', \lambda'')$, seja $\overline{\mathbf{x}}$ uma solução ótima para SL(λ', λ'') e $z(\lambda)$ seu valor correspondente. Denotando por $g \in \mathbb{R}^{|\mathcal{C}' \cup \mathcal{C}''|}$ o vetor de subgradientes no ponto $\bar{\mathbf{x}}$, temos então que

$$g_C = \sum_{i \in C} \overline{x}_i - 1, \text{ para toda clique } C \in \mathcal{C}' \cup \mathcal{C}'' \tag{15}$$

e calculamos um *tamanho de passo* θ, através da fórmula

$$\theta = \frac{\alpha(z(\lambda) - l_{inf})}{\sum_{C \in \mathcal{C}' \cup \mathcal{C}''} g_C^2} \tag{16}$$

onde α é um número real assumindo valores em $(0, 2]$. Feito isso, atualizamos os valores dos multiplicadores λ através da expressão

$$\lambda_i := \max\{0; \lambda_i - \theta g_i\}, i = 1,...,m, \tag{17}$$

recalculamos os custos Lagrangeanos das variáveis $\mathbf{x}$, e passamos para iteração seguinte do MS.

Em nossa implementação do Método do Subgradiente, foi imposto um limite máximo de 10.000 iterações para o algoritmo. O valor do parâmetro α é reduzido à metade sempre que 500 iterações consecutivas são efetuadas sem obter uma redução global no valor do limite superior, l_{sup}. Por outro lado, o MS é interrompido precocemente se l_{inf} e l_{sup} coincidem, fornecendo assim um certificado de otimalidade para a solução viável em mãos. Descrevemos a seguir o procedimento utilizado para dualizar dinamicamente desigualdades válidas num ambiente NDRC. No nosso caso específico, desigualdades de cliques e de buraco ímpar.

3.3 Dualização Dinâmica de Desigualdades Violadas

Sem entrar em maiores detalhes, pode-se dizer que as desigualdades (10) são *permanentes* enquanto as desigualdades (11) são *temporárias*. As primeiras são utilizadas do início ao fim da aplicação do MS. As demais são utilizadas apenas enquanto permanecem *ativas*, ou seja, dualizadas explicitamente e com seus multiplicadores assumindo valores estritamente positivos. Quando o valor de um desses multiplicadores eventualmente cai a zero (na operação de atualização, (17)), sua desigualdade correspondente se torna *inativa* e é então descartada de $\mathcal{C}''$. Note, no entanto, que nada a impede de voltar a fazer parte daquele conjunto, futuramente. Para tanto, basta que volte a ser violada em uma solução subsequente de SL(λ', λ'') e, além disso,

seja identificada como violada pelo algoritmo de *separação* (*identificação*) utilizado. Mais detalhes sobre esses procedimentos serão apresentados ao longo do texto.

Como um todo, o processo descrito acima é extremamente dinâmico e o número de desigualdades em $\mathcal{C}''$ pode variar ao longo da aplicação do MS, com $|\mathcal{C}''|$ aumentando ou diminuindo. À cada iteração do algoritmo, um procedimento heurístico é aplicado a $\bar{\mathbf{x}}$, para identificar desigualdades de clique que não pertencem a $\mathcal{C}' \cup \mathcal{C}''$ e que são violadas naquele ponto. Estas, idealmente, devem ter a maior cardinalidade possível e são então dualizadas, incorporadas à $\mathcal{C}''$, e submetidas à restrição de permanência definida acima. Assim procedendo, o número de desigualdades em $\mathcal{C}''$ tende a se manter relativamente baixo.

Mesmo sob nossa reformulação do PCIM, o número de desigualdades em $\mathcal{C}'$ pode vir a ser *excessivamente alto*. Decidimos então também aplicar a estas desigualdades a mesma regra de dualização dinâmica descrita acima. Ressaltamos que este tratamento não é usual em algoritmos NDRC. Em outras aplicações, desigualdades permanentes são normalmente tratadas de forma estática, por serem relativamente poucas.

Na realidade, a regra utilizada é ligeiramente diferente daquela descrita acima. Exemplificando, quando o multiplicador de Lagrange de uma desigualdade em $\mathcal{C}'$ cai a zero, esta é colocada sob uma forma de quarentena. Não é considerada nas operações de atualização de multiplicadores (15)–(17), mas passa a ser testada para violação à toda iteração subsequente do MS. Tão logo seja violada, volta a fazer parte de $\mathcal{C}'$, como um membro pleno.

Note que não dispensamos às desigualdades de $\mathcal{C}''$ o mesmo tratamento aplicado no parágrafo anterior às desigualdades de $\mathcal{C}'$. Ou seja, uma vez que uma desigualdade de $\mathcal{C}''$ se torne inativa e seja descartada daquele conjunto, não a colocamos num processo de quarentena. Mantemos a desigualdade armazenada em memória (para uso futuro em uma das variantes do algoritmo exato a ser descrito no Capítulo 5). No entanto, para evitar que o conjunto $\mathcal{C}''$ eventualmente cresça em demasia, não sujeitamos a desigualdade à quarentena. Ressaltamos, de qualquer forma, que a aplicação da regra pode vir a ser uma alternativa interessante a testar no futuro.

Outro ponto importante a destacar é que, no caso específico do PCIM, a fronteira que estabelecemos entre desigualdades permanentes e temporárias não é tão rígida. Note que as desigualdades em $\mathcal{C}'$ são necessárias para que tenhamos em mãos uma formulação do problema. No entanto, quanto identificamos dinamicamente uma nova desigualdade de clique, que seja não apenas violada mas que também domina uma ou mais desigualdades do conjunto $\mathcal{C}'$, estas são então descartadas e a nova desigualdade toma os seus lugares no conjunto $\mathcal{C}'$, tornando-se assim permanente. Ao fazer isso, as desigualdades de $\mathcal{C}'$ continuam a definir uma formulação para o PCIM. A nova formulação, no entanto, é mais forte que a anterior. Finalmente,

para dar continuidade suave à aplicação do MS, o valor atribuído ao multiplicador da nova desigualdade é o maior, dentre todos aqueles associados às desigualdades que estão sendo substituídas.

3.3.1 Separação de Cliques

Utilizamos uma heurística para identificar desigualdades de clique violadas no ponto $\overline{\mathbf{x}}$. Esta é composta por uma procedimento construtivo, inicial, complementado por um procedimento de Busca Local (BL). O procedimento construtivo se baseia naquele proposto por NEMHAUSER e SIGISMONDI (1992), para $\overline{\mathbf{x}}$ fracionário. Note, no entanto, que no nosso caso $\overline{\mathbf{x}}$ é binário 0-1.

À cada iteração do MS, construímos o grafo suporte correspondente a $\bar{\mathbf{x}}$, que denotamos por $\bar{G} = (\bar{V}, \bar{E})$. Ou seja, o subgrafo de G induzido pelos vértices $i \in V$ para os quais $\bar{x}_i = 1$. Para inicializar uma clique violada, C, escolhemos o vértice de maior *grau* em $\bar{G}$. Ou seja, aquele com o maior número de arestas a ele incidentes em $\bar{G}$. Assuma que $j \in \overline{V}$ é esse vértice. Eliminamos de $\bar{G}$ todos os vértices que não são vizinhos de j, ou seja, que não possam ser utilizados para expandir a clique C. A seguir, introduzimos em C o vértice de maior grau no grafo resultante e repetimos as operações anteriores. Ao final do processo, obtemos uma clique de $\bar{G}$ (e também de G) que é violada no ponto $\bar{\mathbf{x}}$.

Com uma clique C em mãos, passamos então para um procedimento de BL. Isso é feito testando trocas um-por-dois. Ou seja, excluímos um vértice de C e testamos a inclusão de outros dois a ela. Quando bem sucedidos, efetuamos a troca e reinicializamos todo o procedimento de BL para a nova clique obtida. Caso contrário, o algoritmo prossegue, considerando outro vértice como candidato a trocas. O procedimento termina quando todos os vértices da clique são investigados e nenhuma troca é efetuada. Neste ponto, passamos então a considerar os vértices de $V \setminus \overline{V}$. Testamos se um destes forma uma clique com todos os demais vértices de C. Isto é feito até o ponto em que não seja mais possível expandir C, mesmo com a aplicação do procedimento de BL.

3.4 O SL como um Problema da Mochila 0-1

O problema da mochila 0-1 (PM) [47] é um dos problemas mais investigados e utilizados em aplicações práticas de Otimização Combinatória (veja MARTELLO(1990), para maiores detalhes). Pode ser descrito como em [48], por exemplo, onde temos n itens candidatos a carregar numa mochila de capacidade CAP, cada item j trazendo um *benefício* $w_j \in \mathbb{R}_+$ e pesando $p_j \in \mathbb{Z}$. O objetivo sendo o de escolher um subconjunto de itens que maximize o benefício a obter, sem exceder, no entanto, a

capacidade da mochila. Associando uma variável $x_j \in \{0,1\}$ à decisão de carregar ou não o item j na mochila, o PM pode ser então formulado como:

$$\text{Maximizar} \sum_{j \in n} w_j x_j \tag{18}$$

sujeito a

$$\sum_{j \in n} p_j x_j \leq CAP \tag{19}$$

$$x_j \in 0, 1, j = 1, ..., n \tag{20}$$

Numa tentativa de obter limites duais mais fortes para nossa reformulação do PCIM, testamos a utilização das duas alternativas de SL aqui consideradas. Para nosso algoritmo NDRC, um SL genérico definido por PM é descrito como:

$$\text{Maximizar} \sum_{i \in V} w_i x_i + \ \text{CT}(\lambda', \lambda'') \tag{21}$$

sujeito a

$$\sum_{i \in V} p_i x_i \leq CAP \tag{22}$$

$$x_i \in \{0,1\}, i \in V \tag{23}$$

onde $\sum_{i \in V} p_i x_i \leq CAP$ é a desigualdade obtida ao somarmos todas as desigualdades de clique em C' e C''.

3.4.1 Identificação do Melhor SL a Utilizar

Para identificar a melhor alternativa de SL a utilizar, executamos alguns experimentos computacionais. Em particular, comparamos o desempenho de algoritmos NDRC utilizando cada um dos dois SLs definidos acima. Denominamos **Livre-Escolha** ao algoritmo baseado no SL (13)-(14) e que utiliza as informações dos limites inferiores e superiores globais para resolver o problema. Denominamos **PM**, àquele baseado no problema da mochila definido por (21)-(23).

Para as duas alternativas acima, apresentamos, na Tabela 3.4.1, as seguintes informações: número médio de iterações executadas antes do critério de parada ser atingido (por prova de otimalidade ou 10.000 iterações efetuadas) para cada conjunto de instâncias disponíveis, **ITER**, percentual das vezes em que cada alternativa NDRC obteve melhor desempenho, e o percentual de empates obtidos entre elas.

Conjunto	**INTER**	**Livre-Escolha**	**PM**	**EMPATE**
BROCK	10.000	70,4%	7,5%	22,1%
C	10.000	64,4%	9,2%	26,4%
C-FAT	2.263	30,2%	53,3%	16,5%
DSJC	10.000	38,0%	34,8%	27,2%
GEN	1.266	47,6%	4,4%	48,0%
HAMMING	3.424	36,6%	30,6%	32,8%
JOHNSON	36	1,8%	57,3%	40,9%
KELLER	10.000	91,4%	1,1%	7,5%
MANN	10.000	40,7%	50,6%	8,7%
P-HAT	10.000	60,1%	12,2%	27,7%
SAN	3.943	66,5%	6,1%	27,3%
FRB	9.086	96,3%	0,4%	3,3%

Tabela 3.1: Comparação de desempenho **Livre-Escolha** x **PM**

Como é possível observar, a alternativa **Livre-Escolha** obteve um melhor desempenho neste experimento. Levou vantagem para 63% de todos as instâncias testadas. A alternativa **PM** foi superior em apenas 17,7% dos casos, ocorrendo um empate em 19,3% dos casos restantes.

Comparamos também a performance dos limites inferiores e superiores globais obtidos pelo algoritmo NDRC, em cada caso. Na Tabela 3.2, apresentamos a diferença percentual entre o limite inferior, **LI**, e o limite superior, **LS**, obtidos por **PM**, em relação àqueles obtidos por **Livre-Escolha**. Dessa forma, identificamos a diferença percentual da amplitude alcançada pelas duas alternativas, expressas em **GAP**.

Conjunto	LI	LS	GAP
BROCK	1,5%	2,3%	2,6%
C	0,3%	1,8%	2,2%
C-FAT	0,0%	0,1%	2,9%
DSJC	0,0%	2,0%	2,5%
GEN	0,3%	0,0%	-22,1%
HAMMING	0,0%	0,1%	3,0%
JOHNSON	0,0%	-2,6%	-32,1%
KELLER	1,1%	0,4%	-3,0%
MANN	0,0%	0,0%	0,0%
P-HAT	0,0%	1,2%	2,1%
SAN	0,2%	0,7%	2,9%
FRB	0,1%	0,1%	0,9%
TOTAL	**0,1%**	**0,6%**	**2,0%**

Tabela 3.2: Diferença nos limites primal e dual **PM** em relação ao **Livre-Escolha**

A alternativa **PM** gerou limites superiores 0,6% mais altos (piores). Porém apresentou um melhor desempenho na geração de limites inferiores, da ordem de 0,1%. Nossa decisão pendeu a favor da alternativa **Livre-Escolha**, já que tomamos por critério de escolha a obtenção de melhores limites superiores.

Finalmente, efetuamos um último experimento. Neste, à cada iteração do MS, consideramos tanto a solução obtida pelo SL associado à alternativa **PM** quanto aquela associada à alternativa **Livre-Escolha**, tomando sempre a melhor das duas. De forma geral, essa alternativa híbrida obteve os piores resultados. Com a alternância dos tipos de SL resolvidos à cada iteração, o MS apresentou uma convergência pouco satisfatória. Mantivemos assim nossa opção anterior pela alternativa **Livre-Escolha**.

3.5 Dualização Dinâmica de Buracos Ímpares

Considere, como antes, um grafo $G = (V, E)$. Um *caminho* em G é uma sequência de vértices não repetidos de V, conectados por uma sequência de arestas de E. Um *ciclo* de G, por sua vez, é um caminho formado por três ou mais vértices, acrescido de uma aresta adicional, que conecta suas extremidades. O ciclo é *ímpar* quando contém um número ímpar de vértices e é *induzido* quando inexistem arestas de G ligando pares de vértices não consecutivos do mesmo. Ciclos induzidos ímpares contendo cinco ou mais vértices são chamados alternativamente de *buracos ímpares*.

Numa tentativa de reforçar nossos limites NDRC duais, investigamos, além da dualização dinâmica de desigualdades de clique, a dualização de desigualdades de buraco ímpar. Para tanto, como no caso da separação de cliques, trabalhamos sobre o grafo suporte $\bar{G} = (\bar{V}, \bar{E})$, correspondente a uma solução ótima $\overline{\mathbf{x}}$ de $\mathrm{SL}(\lambda', \lambda'')$. Para identificar buracos ímpares de $\bar{G}$, utilizamos uma adaptação da heurística proposta por FONDZEFE (2016).

A heurística se baseia numa *árvore em camadas*, $T = (\bar{V}_T, \bar{E}_T)$, construída a partir do grafo $\bar{G}$. Em particular, utilizamos a Figura 3.1 para ilustrar o procedimento de construção. O grafo $\bar{G}$ é o que aparece à esquerda da figura. A árvore T, cuja construção vamos descrever, é aquela que aparece à direita da mesma.

Para inicializar T, escolhemos um vértice qualquer para atuar como *raiz*. No nosso caso específico, sem entrar em maiores detalhes, escolhemos o vértice 2 para cumprir esse papel. Tal vértice define então a camada 0 de T. Para expandir T em sua camada 1, consideramos as arestas de $\bar{G}$ que são incidentes a 2. Ou seja, aquelas com extremidades respectivamente nos vértices 1 e 3. Feito isso, T passa a ter duas camadas e repetimos o procedimento anterior para continuar sua expansão. Isso é feito considerando, de forma ordenada, cada um dos vértices da camada 1. Sem entrar em maiores detalhes, vamos assumir que essa ordem é definida pelo vértice 1, seguido do vértice 3. Expandimos T a partir de 1 considerando todas as arestas de $\bar{G}$ incidentes a 1 e que não sejam incidentes a vértices já incluídos em T. No nosso caso específico, apenas a aresta $\{1, 4\}$ se qualifica para a função. Para o vértice 3, por sua vez, a expansão é feita através das arestas $\{3, 5\}$, $\{3, 6\}$ e $\{3, 10\}$. Procedendo recursivamente como acabamos de descrever, chegamos eventualmente à árvore em camadas de $\bar{G}$, ilustrada na Figura 3.1.

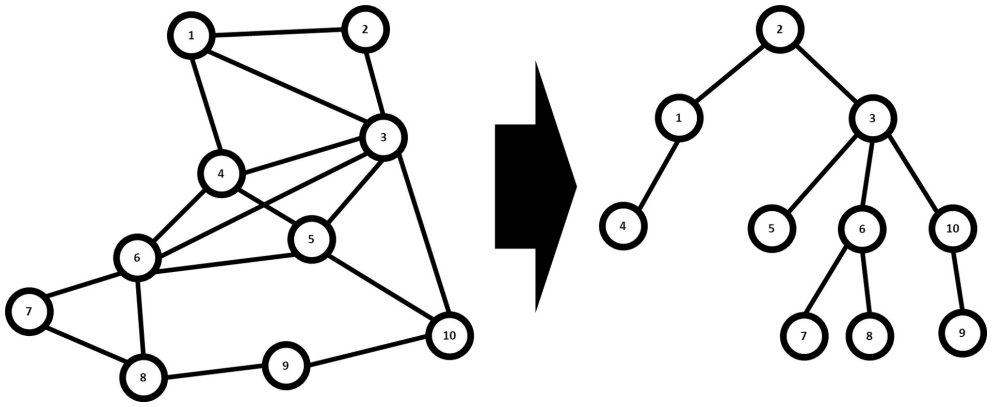

Figura 3.1: Construção da Árvore em Camadas de $\bar{G}$

Para identificar ciclos ímpares de $\bar{G}$, nos bastaria considerar certos pares de vértices localizados numa camada $t \geq 2$ de T. Note que um caminho par conectando tais pares de vértices necessariamente existe naquela árvore e passa por um ancestral em comum. No entanto, para que nos leve a um ciclo ímpar, uma aresta de $\bar{G}$ deve

necessariamente existir entre os dois vértices. Na figura Figura 3.2 consideramos os vértices 4 e 5, localizados na camada 2. Tais vértices se qualificam para a função pois a aresta $\{4, 5\}$ existe entre eles, em $\bar{G}$. Note que combinando o caminho entre os dois com a aresta que os conecta diretamente, chegamos ao ciclo ímpar mostrado na figura. Este, como se pode observar, contém cinco vértices.

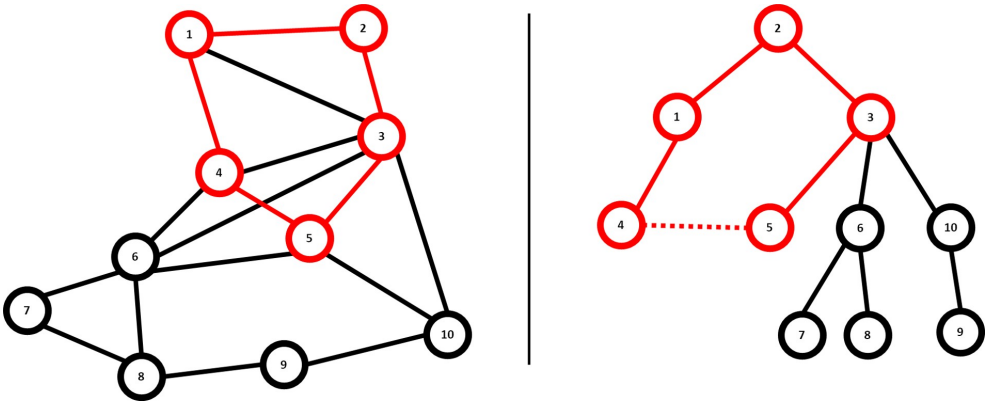

Figura 3.2: Exemplo de Identificação do Ciclo Ímpar (em vermelho)

Identificado um ciclo ímpar, verificamos se este define ou não um buraco ímpar. No exemplo da figura 3.2, o ciclo ímpar U, formado pelos vértices $\{1, 2, 3, 4, 5\}$, não cumpre essa função. Chega-se a essa conclusão através de um procedimento que verifica a existência ou não de arestas internas a U. Se o resultado é positivo, dividimos U em dois ciclos: um necessariamente par e outro necessariamente ímpar. A seguir, tomamos o ciclo ímpar resultante como sendo nosso novo ciclo U e repetimos o procedimento anterior até que este mais se aplique. Neste ponto, se o ciclo ímpar em mãos tiver menos do que 5 vértices, a heurística terá falhado. Caso contrário, um buraco ímpar foi identificado e é denotado por B, sendo a sua desigualdade de buraco ímpar correspondente,

$$\sum_{v \in B} x_v \leq k, \text{ onde } k = \lfloor \frac{|B|}{2} \rfloor, \tag{24}$$

violada em $\overline{\mathbf{x}}$.

A desigualdade (24) não define necessariamente uma faceta do politopo correspondente à envoltória convexa de todos os CIs. Para que isso ocorra, a desigualdade deve ser submetida a um procedimento de *lifting*. Em nossa investigação, aplicamos o procedimento sugerido em [15]. Este identifica parâmetros $\alpha_v \in \mathbb{Z}$, para $v \in V \setminus B$, tais que

$$\sum_{v \in B} x_v + \sum_{v \notin B} \alpha_v x_v \leq k, \tag{25}$$

$0 \leq \alpha_v \leq k$. Valores para os coeficientes α_v dependem da ordem específica em que os vértices do conjunto $V \setminus B$ são investigados. Em particular, se $V \setminus B = \{v_1, v_2, ..., v_r\}$,

$r = |V| - (2k + 1)$, e os coeficientes são calculados na ordem de α_{v_i}, $i = 1, ..., r$, temos então que

$$\alpha_{v_j} = k - z_{v_j} \tag{26}$$

onde z_{v_j} é definido como

$$z_{v_j} = \text{máximo} \sum_{v \in B} + \sum_{i=1}^{j-1} \alpha_{v_i} x_{v_i}. \tag{27}$$

Note, nesse caso, que a maximização é feita resolvendo o PCIM definido pelo conjunto de vértices $(B \cup \{v_1, ..., v_{j-1}\}) \setminus \Gamma(v_j)$, onde $\Gamma(v)$ representa o conjunto dos vizinhos do vértice v. Para maiores detalhes sobre o procedimento, sugerimos uma consulta a [15].

3.5.1 O Impactos Observados Com a Dualização de Desigualdades de Buraco Ímpar

Observamos um aumento no tempo de CPU, provocado pelo uso de desigualdades de buraco ímpar, da ordem de 69%, em média. No entanto, não observamos ganhos significativos na qualidade dos limites primais e duais que obtivemos. Para apenas 35% das instâncias testadas conseguimos melhorar, de forma marginal, o desempenho do algoritmo. Especificamente, o impacto obtido para elas foi de uma redução média de apenas 0,4%, no limite dual.

Na média de todas as instâncias de teste consideradas, a utilização de desigualdades de buraco ímpar levou a um aumento no limite superior da ordem de 0,3%, ou seja, a uma piora. Dessa maneira, descartamos então a utilização dessas desigualdades no algoritmo NDRC. Futuramente, no entanto, pretendemos voltar a investigá-las.

Capítulo 4

Heurística Primal

Neste capítulo, apresentamos a heurística primal que utilizamos para gerar soluções viáveis para o PCIM. Em particular, mostramos como essa heurística se beneficia da informação dual obtida à cada iteração do MS. Após a aplicação da heurística, submetemos a solução resultante a um procedimento de BL, baseado em Busca Tabu, como uma forma de tentar fugir de ótimos locais.

Abaixo apresentamos brevemente o passo a passo da heurística, que será mais detalhada a seguir. Considere D uma lista de vértices inicialmente vazia, a solução lagrangeana e $\bar{G}$ o subgrafo induzido por .

1. Tome i com o melhor custo-benefício e $D = D \cup i$
2. Elimine todos os vizinhos de i de $\overline{G}$
3. Repita os passos 1 e 2, até que $\overline{G} = \emptyset$
4. Faça o mesmo procedimento com os vértices restantes de G
5. E, ao final, usamos uma Busca Local com condição TABU para procurar uma melhora na solução em mãos.

4.1 Algoritmo Construtivo para Solução Viável

Como antes, seja $\bar{\mathbf{x}}$ uma solução ótima para o SL(λ', λ'') e $\bar{G} = (\bar{V}, \bar{E})$ o seu grafo suporte (induzido por $G(\bar{V})$). Denominamos por $D = \emptyset$ o conjunto independente a ser gerado e, para inicializá-lo, escolhemos $i \in \overline{V}$, sujeito ao critério do melhor custo-benefício $\frac{w_i}{|\overline{\Gamma}(i)|}$, onde w_i é o custo Lagrangeano de x_i e $\overline{\Gamma}(i)$ é a vizinhança aberta de i em $\overline{G}$. Feito isso, o vértice i é eliminado de $\bar{G}$, juntamente com todos os seus vizinhos, ou seja, $\overline{\Gamma}(i)$. Para o grafo reduzido resultante, selecionamos, mais uma vez, o vértice com o melhor custo-benefício para expandir D. Da mesma forma, eliminamos do mesmo, a seguir, aquele vértice e seus vizinhos. Este procedimento

básico é repetido até que o grafo em mãos se torne vazio. Neste ponto, excluímos de G os vértices de D, assim como todos os seus vizinhos. A seguir, passamos a aplicar ao grafo resultante, o procedimento de expansão de D descrito acima para $\bar{G}$. Mais uma vez, isso é feito até que o grafo em mãos se torne vazio.

Como é possível verificar, nossa heurística construtiva envolve duas etapas. A primeira, privilegia os vértices de $\bar{G}$, ou seja, utiliza a informação dual Lagrangeana. A segunda, tenta expandir D, concentrando-se nos demais vértices de G. No entanto, por se tratar de uma heurística gulosa, o algoritmo tende a nos levar para um ótimo local. Para tentar contornar essa limitação, aplicamos a D o procedimento de BL mencionado na introdução desta seção. Apresentamos a seguir, no Algoritmo 2, um pseudo-código para nossa heurística gulosa.

Algorithm 2 Heurística Gulosa para Solução Viável

Require: $G = (V, E)$ and $\bar{G} = (\bar{V}, \bar{E})$ a set of vertex $D := \emptyset$
Ensure: a independent set D
  AUX $\leftarrow \{\}$
  while $|\bar{V} \setminus \text{AUX}| > 0$ **do**
    $v^* \leftarrow v_k \in (\bar{V} \setminus \text{AUX})$, such the $\frac{w_k}{|\bar{\Gamma}(k)|}$ is maximal
    $D \leftarrow D \cup \{v^*\}$
    AUX $\leftarrow$ AUX $\cup \{v^* \ \cup \Gamma(v^*)\}$
  end while
  while $|V \setminus \text{AUX}| > 0$ **do**
    $v^* \leftarrow v_k \in (V \setminus \text{AUX})$, such the $\frac{w_k}{|\bar{\Gamma}(k)|}$ is maximal
    $D \leftarrow D \cup \{v^*\}$
    AUX $\leftarrow$ AUX $\cup \{v^* \ \cup \Gamma(v^*)\}$
  end while
  return D

4.2 Busca TABU

Uma vez concluído o procedimento descrito acima, uma BL é aplicada a D. O objetivo sendo o de tentar aumentar a cardinalidade daquele conjunto. Isso é feito utilizando elementos de Busca Tabu [50] e a lógica de trocas empregada por ANDRADE *et al.* [13], ou seja de trocas 1-por-2 ou 2-por-3. Permitimos um máximo de 200 iterações para o procedimento. Além disso, fixamos os parâmetros α e β que vamos utilizar em $\alpha = 25$ e $\beta = 15$, onde β é multiplicado por um número aleatório entre 0 e 1, para gerar um índice $\mathcal{T} = \alpha + \beta * \text{rand}()$, que representa o número de iterações consecutivas em que um vértice excluído de D não pode voltar àquele conjunto. Isso é feito, atualizando uma lista TABU $\mathcal{T}$ de tamanho $|V|$, para impor

que o vértice excluído permanecerá entre 25 e 40 iterações das 200 possíveis sem poder ser candidato a entrar, ou sair, novamente de D. Note que a condição TABU de um vértice é imposta como uma tentativa de fugir de ótimos locais, possibilitando a entrada de vértices com custo benefício menos atrativo pela heurística gulosa.

Seja $i \in D$, numa dada iteração do algoritmo, o vértice escolhido para exclusão. Para substituí-lo, identificamos então o maior número possível de vértices na vizinhança aberta de $D \setminus \{i\}$, ou seja, $\Gamma(D \setminus \{i\})$, não impedidos (temporariamente) de entrar em D. Isso é feito aplicando essencialmente o mesmo procedimento indicado acima para a construção de D. Consideramos todos os vértices não-TABU existentes em D, e escolhemos aquele que leva à troca mais vantajosa, na iteração corrente da BL. Durante todo o processo de troca, guardamos a melhor solução encontrada. Como saída do procedimento de BL teremos, possivelmente, uma melhor solução. No pior cenário, a mesma encontrada anteriormente pela heurística construtiva. Apresentamos a seguir um pseudo-código para o procedimento descrito acima.

Algorithm 3 Busca Local TABU para Solução Viável

Require: $G = (V, E)$, solution S, α, β and Max_{it}

Ensure: solution S_{best} hopefully better

```
n ← |V|
T ← array(n)
S_best ← S
for i = 1, ..., n do
    T[i] ← 0
end for
for it = 1, ..., Max_it do
    S_size ← 0
    S'_best ← 0
    for vertice i ∈ S do
        S' ← S \ {i}
        S' ← expand_solution(S', i, T)
        if |S'| > S_size then
            S_size ← |S'|
            S'_best ← S'
        end if
    end for
    for vertice i ∈ (S ∪ S'_best) \ (S ∩ S'_best) do
        T[i] ← T[i] + α + β * rand(0, 1)
    end for
    S ← S'_best
    if z(S) > z(S_best) then
        S_best ← S
    end if
end for
return S_best
```

Algorithm 4 Procedimento de Expansão da Solução Viável

```
function expand_solution(S, v_out, T) {
    Res_S ← S
    for i ∈ V \ (S ∪ Γ(S)) do
        if (i ≠ v_out) and (T[i] ≤ it) then
            if (Res_S ∪ i) is a valid solution then
                Res_S ← Res_S ∪ {i}
            end if
        end if
    end for
    return Res_S
}
```

Capítulo 5

Resultados Computacionais

Apresentamos inicialmente os resultados computacionais obtidos com a aplicação do algoritmo NDRC para a resolução do PCIM. As instâncias de teste utilizadas são aquelas mencionadas previamente e a versão utilizada do algoritmo dualiza apenas as desigualdades de clique. A seguir, apresentamos os resultados obtidos ao combinarmos o algoritmo NDRC com o software CPLEX. Neste caso, o algoritmo NDRC atua como um *warm starter* para uma inicialização avançada do CPLEX. Denominamos NDRC-CPLEX a este algoritmo híbrido.

Todos os experimentos computacionais foram efetuados em um computador equipado com um processador Intel i7-7700k, com uma velocidade de 4,2 GHz e 32 GB de RAM. Nossos códigos foram escritos na linguagem de programação C++, o software de Programação Inteira utilizado foi o IBM ILOG CPLEX na versão 12.7.1 [9] e o sistema operacional empregado foi o Windows 10 Pro 64 bits.

A Tabela 5.1 descreve os resultados obtidos pelo algoritmo NDRC. Estes aparecem em colunas rotuladas como: (a) **Total**, indica o número de instâncias existentes em cada conjunto considerado, (b) **LI-opt**, indica o número de instâncias para as quais o limite inferior obtido foi igual ao valor ótimo (ou igual ao melhor limite inferior conhecido, quando o valor ótimo é desconhecido), (c) **LS-opt**, indica o número de instâncias para as quais o limite superior foi igual ao valor ótimo (ou igual ao melhor limite inferior conhecido, quando o valor ótimo é desconhecido), (d) **(LI/LS)-opt**, o número de instâncias para as quais o limite inferior ou o limite superior foram iguais ao valor ótimo (ou iguais ao melhor limite inferior conhecido, quando o valor ótimo é desconhecido), (e) **OPT**, o número de instâncias para as quais um certificado de otimalidade foi obtido, e (f) **Tempo**, o tempo médio de CPU "de parede", em segundos, gastos para resolver/tentar resolver as instâncias de cada grupo.

Grupo	Total	LI-opt	LS-opt	(LI/LS)-opt	OPT	Tempo(seg)
BROCK	12	6	0	6	0	77
C	7	4	0	4	0	426
C-FAT	7	7	6	7	6	17
DSJC	2	2	0	2	0	114
GEN	5	4	5	5	4	20
HAMMING	6	6	4	6	4	54
JOHNSON	4	4	4	4	4	4
KELLER	3	2	0	2	0	320
MANN	4	2	0	2	0	2237
P-HAT	13	13	0	13	0	173
SAN	15	14	11	15	10	17
FRB	41	6	41	41	6	276
TOTAL	**119**	**70**	**71**	**107**	**34**	**311**

Tabela 5.1: Resultados Obtidos pelo Algoritmo NDRC

No total, para 107 das 119 instâncias consideradas, o valor ótimo foi alcançado através do limite inferior ou do limite superior. Conseguimos obter também certificados de otimalidade para 34 instâncias.

5.1 Algoritmo NDRC-CPLEX

Descrevemos agora os resultados obtidos pelo algoritmo NDRC-CPLEX. Este é empregado sempre que o algoritmo NDRC, por si só, não é capaz de resolver à otimalidade uma dada instância do problema. Embora a aplicação do procedimento tenha sido exigida para apenas 85 das 119 instâncias consideradas, incluímos na Tabela 5.2 os resultados obtidos para as 34 instâncias restantes, resolvidas à otimalidade exclusivamente pelo algoritmo NDRC.

Para instâncias que de fato demandam a aplicação do NDRC-CPLEX, o algoritmo NDRC funciona como um *warm starter*. Ou seja, como um instrumento capaz de viabilizar uma inicialização avançada do CPLEX. Isso é feito fornecendo àquele software, além de nossa reformulação para o PCIM, as desigualdades dualizadas dinamicamente que se encontravam ativas no momento em que o melhor limite dual foi obtido pelo algoritmo NDRC. Também fornecemos ao CPLEX a melhor solução viável obtida ao longo da aplicação do algoritmo NDRC. Assim procedendo este tem, como ponto de partida, o melhor limite dual e o melhor limite primal obtidos pelo algoritmo NDRC.

As informações relativas ao algoritmo NDRC-CPLEX que aparecem na Tabela 5.2 são similares àquelas que a Tabela 5.1 fornece para o algoritmo NDRC. Uma diferença, no entanto, diz respeito ao tempo de CPU, que agora considera tanto o tempo dispendido pelo algoritmo NDRC quanto aquele relativo ao uso do CPLEX. Um limite de 7.200 segundos de tempo de CPU "de parede"foi imposto em cada rodada do CPLEX.

Grupo	**Total**	**LI-opt**	**LS-opt**	**(LI/LS)-opt**	**OPT**	**Tempo(seg)**
BROCK	12	7	4	7	4	4891
C	7	4	1	4	1	6600
C-FAT	7	7	7	7	7	17
DSJC	2	2	0	2	0	7317
GEN	5	5	5	5	5	20
HAMMING	6	6	5	6	5	1254
JOHNSON	4	4	4	4	4	4
KELLER	3	2	1	2	1	5120
MANN	4	4	4	4	4	2378
P-HAT	13	13	7	13	7	4308
SAN	15	15	14	15	14	762
FRB	41	12	41	41	12	5505
TOTAL	**119**	**81**	**93**	**110**	**64**	**3181**

Tabela 5.2: Resultados Obtidos pelo Algoritmo NDRC-CPLEX

O algoritmo NDRC-CPLEX resolveu à otimalidade 64 das nossas 119 instâncias de teste (34 destas através da aplicação direta do algoritmo NDRC). Além disso, conseguiu obter certificados de otimalidade para 3 das 18 instâncias que ainda se encontravam em aberto, algumas há mais de 30 anos. Especificamente, comprovamos que os melhores limites inferiores previamente obtidos para as instâncias **C.125.9**, **MANN-a81** e **p-hat700-2** correspondem, de fato, aos valores ótimos para as mesmas, respectivamente de 34, 1.100 e 44.

Nas Tabelas 5.3 e 5.4, apresentamos resultados computacionais detalhados para cada uma das 119 instâncias de teste aqui consideradas. Os resultados se estendem por 13 colunas, divididas em três grupos. O primeiro grupo fornece informações sobre os grafos de definição, $G = (V, E)$, correspondentes à cada instância de teste. Indica também seus valores ótimos (ou os melhores limites inferiores conhecidos para cada instância, quando o ótimo é desconhecido). Da mesma forma, indica ainda os valores da relaxação linear da formulação padrão, aplicada à cada instância. O segundo e o terceiro grupos estão associados à aplicação dos algoritmos NDRC e

NDRC-CPLEX, operando sob nossa reformulação do PCIM. Mais especificamente, temos os seguintes rótulos para as colunas:

- **NOME**: identifica a instância;
- **N**: Número de variáveis na formulação padrão do PCIM, que é igual ao número de vértices de G;
- **M**: Número de restrições na formulação padrão do PCIM, que é igual ao número de arestas de G;
- **BEST**: Valor da solução ótima ou do melhor limite inferior conhecido, quando o ótimo é desconhecido (destacado em vermelho quando a instância ainda se encontra em aberto);
- **RL**: Valor da relaxação linear da formulação padrão do PCIM;
- **LI**: Melhor limite inferior encontrado por NDRC ou NDRC-CPLEX, o que se aplicar;
- **LS**: Melhor limite superior encontrado por NDRC ou NDRC-CPLEX o que se aplicar;
- **GAP**: Diferença percentual entre LI e LS, para NDRC ou NDRC-CPLEX. "OPT"indica que encontramos o valor ótimo;
- **TEMPO**: Tempo "de parede" de CPU em segundos.

NOME	N	M	BEST	RL	NDRC				NDRC-CPLEX			
					LI	LS	GAP	TEMPO	LI	LS	GAP	TEMPO
brock200_1	200	5.066	21	100	21	38,83	17	17	21	21,00	OPT	89
brock200_2	200	10.024	12	100	12	21,93	9	13	12	12,00	OPT	24
brock200_3	200	7.852	15	100	15	28,03	13	14	15	15,00	OPT	42
brock200_4	200	6.811	17	100	17	31,46	14	15	17	17,00	OPT	50
brock400_1	400	20.077	27	200	25	66,44	41	54	25	42,95	17	7.264
brock400_2	400	20.014	29	200	29	66,62	37	54	29	42,34	13	7.261
brock400_3	400	20.119	31	200	31	66,51	35	54	31	41,12	10	7.255
brock400_4	400	20.035	33	200	25	67,24	42	53	33	39,84	6	7.254
brock800_1	800	112.095	23	400	21	101,07	80	162	21	75,32	54	7.363
brock800_2	800	111.434	24	400	21	100,46	79	163	21	76,51	55	7.365
brock800_3	800	112.267	25	400	22	100,54	78	162	22	74,85	52	7.362
brock800_4	800	111.957	26	400	21	101,87	80	163	21	75,51	54	7.364
C1000.9	1.000	49.421	68	500	66	225,14	159	307	66	201,06	135	7.507
C125.9	125	787	34	63	34	43,17	9	12	34	34,00	OPT	12
C2000.5	2.000	999.164	16	1.000	16	167,32	151	450	16	157,70	141	7.650
C2000.9	2.000	199.468	77	1.000	73	411,79	338	829	73	394,47	321	8.029
C250.9	250	3.141	44	125	44	71,90	27	33	44	49,20	5	7.237
C4000.5	4.000	3.997.732	18	2.000	17	312,55	295	1.250	17	302,07	285	8.450
C500.9	500	12.418	57	250	57	125,26	68	103	57	100,03	43	7.313
c-fat200-1	200	18.366	12	100	12	12,94	OPT	2	12	12,00	OPT	2
c-fat200-2	200	16.665	24	100	24	24,88	OPT	< 0	24	24,00	OPT	< 0
c-fat200-5	200	11.427	58	100	58	66,50	8	62	58	58,00	OPT	62
c-fat500-1	500	120.291	14	250	14	15,00	OPT	17	14	14,00	OPT	17
c-fat500-10	500	78.123	126	250	126	126,90	OPT	2	126	126,00	OPT	2
c-fat500-2	500	115.611	26	250	26	26,90	OPT	31	26	26,00	OPT	31
c-fat500-5	500	101.559	64	250	64	64,96	OPT	3	64	64,00	OPT	4
DSJC1000_5	1.000	249.674	15	500	15	91,69	76	171	15	84,54	69	7.371
DSJC500_5	500	62.126	13	250	13	48,47	35	56	13	23,78	10	7.263
frb100-40	4.000	572.774	100	2.000	91	100,03	9	2.549	91	100,00	9	9.749
frb30-15-1	450	17.827	30	225	30	30,92	OPT	2	30	30,00	OPT	2
frb30-15-2	450	17.874	30	225	30	30,66	OPT	7	30	30,00	OPT	7
frb30-15-3	450	17.809	30	225	30	30,01	OPT	32	30	30,00	OPT	32
frb30-15-4	450	17.831	30	225	30	30,46	OPT	9	30	30,00	OPT	9
frb30-15-5	450	17.794	30	225	30	30,53	OPT	1	30	30,00	OPT	1
frb35-17-1	595	27.856	35	298	33	35,00	2	87	35	35,00	OPT	476
frb35-17-2	595	27.847	35	298	34	35,00	1	90	35	35,00	OPT	1.116
frb35-17-3	595	27.931	35	298	35	35,01	OPT	58	35	35,00	OPT	58
frb35-17-4	595	27.842	35	298	34	35,00	1	87	35	35,00	OPT	267
frb35-17-5	595	28.143	35	298	34	35,00	1	90	35	35,00	OPT	201
frb40-19-1	760	41.314	40	380	39	40,21	1	129	40	40,00	OPT	225
frb40-19-2	760	41.263	40	380	39	40,00	1	136	40	40,00	OPT	3.601
frb40-19-3	760	41.095	40	380	38	40,00	2	137	38	40,00	2	7.348
frb40-19-4	760	41.605	40	380	39	40,00	1	140	39	40,00	1	7.340
frb40-19-5	760	41.619	40	380	39	40,00	1	140	39	40,00	1	7.340
frb45-21-1	945	59.186	45	473	43	45,00	2	194	43	45,00	2	7.414
frb45-21-2	945	58.624	45	473	43	45,00	2	194	43	45,00	2	7.416
frb45-21-3	945	58.245	45	473	43	45,00	2	190	44	45,00	1	7.390
frb45-21-4	945	58.549	45	473	43	45,00	2	188	43	45,00	2	7.397
frb45-21-5	945	58.579	45	473	43	45,00	2	195	44	45,00	1	7.395
frb50-23-1	1.150	80.072	50	575	47	50,00	3	256	47	50,00	3	7.477
frb50-23-2	1.150	80.851	50	575	48	50,00	2	265	48	50,00	2	7.471
frb50-23-3	1.150	81.068	50	575	47	50,00	3	272	47	50,00	3	7.498
frb50-23-4	1.150	80.258	50	575	48	50,00	2	263	48	50,00	2	7.468
frb50-23-5	1.150	80.035	50	575	48	50,00	2	262	48	50,00	2	7.474
frb53-24-1	1.272	94.227	53	636	50	53,00	3	302	50	53,00	3	7.518
frb53-24-2	1.272	94.289	53	636	51	53,00	2	313	51	53,00	2	7.524
frb53-24-3	1.272	94.127	53	636	50	53,00	3	309	50	53,00	3	7.524
frb53-24-4	1.272	94.308	53	636	50	53,00	3	308	50	53,00	3	7.516
frb53-24-5	1.272	94.226	53	636	50	53,00	3	310	50	53,00	3	7.521

Tabela 5.3: Resultados individuais (Parte 1)

NOME	N	M	BEST	RL	NDRC				NDRC-CPLEX			
					LI	LS	GAP	TEMPO	LI	LS	GAP	TEMPO
frb56-25-1	1.400	109.676	56	700	53	56,24	3	350	53	56,00	3	7.552
frb56-25-2	1.400	109.401	56	700	53	56,00	3	343	53	56,00	3	7.560
frb56-25-3	1.400	109.379	56	700	53	56,00	3	359	53	56,00	3	7.590
frb56-25-4	1.400	110.038	56	700	53	56,00	3	362	53	56,00	3	7.575
frb56-25-5	1.400	109.601	56	700	53	56,00	3	359	53	56,00	3	7.568
frb59-26-1	1.534	126.555	59	767	56	59,00	3	412	56	59,00	3	7.627
frb59-26-2	1.534	126.163	59	767	56	59,00	3	399	56	59,00	3	7.603
frb59-26-3	1.534	126.082	59	767	55	59,00	4	400	55	59,00	4	7.601
frb59-26-4	1.534	127.011	59	767	56	59,00	3	410	56	59,00	3	7.635
frb59-26-5	1.534	125.982	59	767	55	59,62	4	403	55	59,00	4	7.614
gen200_p0.9_44	200	1.990	44	216	44	44,98	OPT	< 0	44	44,00	OPT	< 0
gen200_p0.9_55	200	1.990	55	100	55	55,99	OPT	18	55	55,00	OPT	18
gen400_p0.9_55	400	7.980	55	220	54	55,00	1	65	55	55,00	OPT	66
gen400_p0.9_65	400	7.980	65	222	65	65,83	OPT	4	65	65,00	OPT	4
gen400_p0.9_75	400	7.980	75	200	75	75,89	OPT	12	75	75,00	OPT	12
hamming10-2	1.024	5.120	512	512	512	512,00	OPT	75	512	512,00	OPT	75
hamming10-4	1.024	89.600	40	512	40	51,20	11	236	40	48,00	8	7.436
hamming6-2	64	192	32	32	32	32,00	OPT	< 0	32	32,00	OPT	< 0
hamming6-4	64	1.312	4	32	4	5,00	1	3	4	4,00	OPT	3
hamming8-2	256	1.024	128	128	128	128,00	OPT	5	128	128,00	OPT	5
hamming8-4	256	11.776	16	128	16	17,00	OPT	2	16	16,00	OPT	2
johnson16-2-4	120	1.680	8	60	8	8,88	OPT	< 0	8	8,00	OPT	< 0
johnson32-2-4	496	14.880	16	248	16	16,96	OPT	14	16	16,00	OPT	14
johnson8-2-4	28	168	4	14	4	4,89	OPT	< 0	4	4,00	OPT	< 0
johnson8-4-4	70	560	14	35	14	14,95	OPT	< 0	14	14,00	OPT	< 0
keller4	171	5.100	11	86	11	14,35	3	7	11	11,00	OPT	9
keller5	776	74.710	27	388	27	31,01	4	79	27	30,90	3	7.279
keller6	3.361	1.026.582	59	1.681	53	63,00	10	873	53	63,00	10	8.073
MANN_a27	378	702	126	189	126	135,00	9	95	126	126,00	OPT	95
MANN_a45	1.035	1.980	345	518	344	360,02	16	746	345	345,00	OPT	751
MANN_a81	3.321	6.480	1.100	1.661	1.098	1.134,01	36	8.104	1.100	1.100,00	OPT	8.662
MANN_a9	45	72	16	23	16	18,00	2	2	16	16,00	OPT	2
p_hat1000-1	1.000	377.247	10	500	10	48,53	38	110	10	28,49	18	7.312
p_hat1000-2	1.000	254.701	46	500	46	109,68	63	444	46	80,94	34	7.644
p_hat1000-3	1.000	127.754	68	500	68	165,59	97	428	68	140,49	72	7.629
p_hat1500-1	1.500	839.327	12	750	12	75,86	63	212	12	58,33	46	7.414
p_hat300-1	300	33.917	8	150	8	17,55	9	19	8	8,00	OPT	58
p_hat300-2	300	22.922	25	150	25	37,90	12	47	25	25,00	OPT	81
p_hat300-3	300	11.460	36	150	36	56,37	20	46	36	36,00	OPT	299
p_hat500-1	500	93.181	9	250	9	27,24	18	38	9	9,00	OPT	376
p_hat500-2	500	61.804	36	250	36	58,72	22	149	36	36,00	OPT	1.413
p_hat500-3	500	30.950	50	250	50	89,63	39	135	50	63,86	13	7.342
p_hat700-1	700	183.651	11	350	11	36,77	25	66	11	11,00	OPT	2.081
p_hat700-2	700	122.922	44	350	44	82,70	38	290	44	44,00	OPT	6.894
p_hat700-3	700	61.640	62	350	62	121,16	59	263	62	98,66	36	7.465
san1000	1.000	249.000	15	500	14	15,22	1	34	15	15,00	OPT	120
san200_0.7_1	200	5.970	30	226	30	30,92	OPT	1	30	30,00	OPT	1
san200_0.7_2	200	5.970	18	228	18	18,99	OPT	< 0	18	18,00	OPT	< 0
san200_0.9_1	200	1.990	70	230	70	70,95	OPT	< 0	70	70,00	OPT	< 0
san200_0.9_2	200	1.990	60	232	60	60,93	OPT	8	60	60,00	OPT	8
san200_0.9_3	200	1.990	44	100	44	44,95	OPT	2	44	44,00	OPT	2
san400_0.5_1	400	39.900	13	236	13	13,99	OPT	1	13	13,00	OPT	1
san400_0.7_1	400	23.940	40	238	40	40,98	OPT	4	40	40,00	OPT	4
san400_0.7_2	400	23.940	30	240	30	30,97	OPT	2	30	30,00	OPT	2
san400_0.7_3	400	23.940	22	242	22	22,01	OPT	14	22	22,00	OPT	14
san400_0.9_1	400	7.980	100	200	100	101,00	OPT	62	100	100,00	OPT	62
sanr200_0.7	200	6.032	18	246	18	34,00	15	15	18	18,00	OPT	69
sanr200_0.9	200	2.037	42	100	42	60,37	18	24	42	42,00	OPT	98
sanr400_0.5	400	39.816	13	250	13	40,00	26	39	13	13,00	OPT	3.789
sanr400_0.7	400	23.931	21	200	21	60,42	39	52	21	36,48	15	7.261

Tabela 5.4: Resultados individuais (Parte 2)

Como é possível observar, as colunas referentes ao algoritmo NDRC-CPLEX apresentam sempre resultados melhores ou iguais àqueles do algoritmo NDRC. Isso se deve ao fato do CPLEX ser sempre inicializado pelo melhor limite superior obtido pelo algoritmo NDRC. É importante também frisar que o tempo referente a NDRC-

CPLEX incorpora o tempo do NDRC. Consequentemente, é sempre maior ou igual àquele (igual, quando o algoritmo NDRC encontra o ótimo).

O GAP obtido foi, em geral, relativamente baixo. Como exceção, temos apenas algumas poucas instâncias. Por exemplo, algumas daquelas encontradas nos grupos **BROCK** e **C**. Quase invariavelmente, encontramos limites inferiores ou limites superiores que se igualaram a valores ótimos correspondentes para as instâncias. Ainda assim, para diversas delas, o CPLEX se mostrou incapaz de fechar GAPs de apenas 1 ou 2 unidades. Em especial, as instâncias do grupo **FRB** se mostraram particularmente difíceis em termos de limites inferiores. Por outro lado, encontramos relaxações lineares com valores iguais aos de suas soluções ótimas inteiras, para todas as instâncias do grupo.

Através de nossa reformulação do PCIM (vide o Capítulo 2), o algoritmo NDRC conseguiu processar a instância **C4000.5** (que originalmente envolvia 4 milhões de restrições sob a formulação padrão), em apenas 20 minutos de tempo. Na média, o tempo gasto para cada instância de teste foi de apenas 4 minutos (sujeito a um número máximo de 10.000 iterações do MS).

A Tabela 5.5 envolve apenas as 18 instâncias que se encontravam em aberto antes de nosso estudo. Mesmo sendo instâncias difíceis, só não encontramos o valor do melhor limite conhecido para apenas cinco delas. Em particular, para a instância **C125.9**, conseguimos chegar a uma prova de otimalidade em apenas 12 segundos de tempo de CPU "de parede".

Tabela 5.5: Resultados para instâncias em aberto

					NDRC				NDRC-CPLEX			
NOME	N	M	BEST	RL	LI	LS	GAP	TEMPO	LI	LS	GAP	TEMPO
C1000.9	1.000	49.421	68	500	66	225,14	159	307	66	201,06	135	7.507
C125.9	125	787	34	63	34	43,17	9	12	34	34,00	OPT	12
C2000.5	2.000	999.164	16	1.000	16	167,32	151	450	16	157,70	141	7.650
C2000.9	2.000	199.468	77	1.000	73	411,79	338	829	73	394,47	321	8.029
C250.9	250	3.141	44	125	44	71,90	27	33	44	49,20	5	7.237
C4000.5	4.000	3.997.732	18	2.000	17	312,55	295	1.250	17	302,07	285	8.450
C500.9	500	12.418	57	250	57	125,26	68	103	57	100,03	43	7.313
DSJC1000_5	1.000	249.674	15	500	15	91,69	76	171	15	84,54	69	7.371
DSJC500_5	500	62.126	13	250	13	48,47	35	56	13	23,78	10	7.263
keller6	3.361	1.026.582	59	1.681	53	63,00	10	873	53	63,00	10	8.073
MANN_a81	3.321	6.480	1.100	1.661	1.098	1.134,01	36	8.104	1.100	1.100,00	OPT	8.662
p_hat1000-1	1.000	377.247	10	500	10	48,53	38	110	10	28,49	18	7.312
p_hat1000-2	1.000	254.701	46	500	46	109,68	63	444	46	80,94	34	7.644
p_hat1000-3	1.000	127.754	68	500	68	165,59	97	428	68	140,49	72	7.629
p_hat1500-1	1.500	839.327	12	750	12	75,86	63	212	12	58,33	46	7.414
p_hat500-3	500	30.950	50	250	50	89,63	39	135	50	63,86	13	7.342
p_hat700-2	700	122.922	44	350	44	82,70	38	290	44	44,00	OPT	6.894
p_hat700-3	700	61.640	62	350	62	121,16	59	263	62	98,66	36	7.465

Fechando esse capítulo, apresentamos os resultados que obtivemos para as 64 instâncias que conseguimos resolver à otimalidade. Na Tabela 5.6, indicamos o desempenho obtido pelo algoritmo NDRC-CPLEX para os grupos de instâncias às

quais elas pertencem. As informações apresentadas são as seguintes: (a) **Ótimos**, indica o número de instâncias para as quais encontramos a melhor solução disponível, (b) **t-NDRC**, o tempo em segundos dispendido pelo algoritmo NDRC, (c) **t-CPLEX**, o tempo em segundos dispendido apenas pelo CPLEX e (d) **t-TOTAL**, o tempo total em segundos dispendido pelo algoritmo NDRC-CPLEX. Na média, o algoritmo completo levou somente 490 segundos (cerca de 8 minutos) para provar otimalidade dessas 64 instâncias.

Família	**Ótimos**	**t-NDRC**	**t-CPLEX**	**t-TOTAL**
BROCK	4	14	36	51
C	1	12	0	12
C-FAT	7	16	0	16
DSJC	0	na	na	na
GEN	5	19	0	20
HAMMING	5	17	0	17
JOHNSON	4	3	0	3
KELLER	1	7	2	9
MANN	4	2236	140	2377
P-HAT	7	93	1506	1600
SAN	14	14	283	297
FRB	12	60	438	499
TOTAL	**64**	**170**	**320**	**490**

Tabela 5.6: Provas de otimalidade obtidas pelo algoritmo NDRC-CPLEX

Capítulo 6

Conclusões e Sugestões de Trabalhos Futuros

Este trabalho desenvolveu e testou computacionalmente um algoritmo do tipo Non Delayed Relax-and-Cut para resolver o Problema do Conjunto Independente Máximo. O algoritmo teve como ponto de partida uma reformulação do problema, obtida através de uma cobertura por cliques das arestas de seu grafo de definição, $G = (V, E)$. Comparada à formulação padrão do problema, nossa reformulação se mostrou muito mais forte, além de requerer, tipicamente, um número muito menor de desigualdades. Ela se configura não apenas como um contribuição original, mas é também extremamente simples. É ainda, possivelmente, a maior contribuição individual desta dissertação. Como exemplo, considere as 119 instâncias de teste que utilizamos em nossa investigação. Sob a formulação padrão do problema, a diferença percentual média obtida entre valores ótimos (ou de melhores limites inferiores conhecidos, para as instâncias ainda em aberto) e limites de relaxação linear, foi de 1.075%. Para nossa reformulação, essa média caiu a 100%, ou seja, uma redução da ordem de 90%. Investigar procedimentos que nos levem a encontrar sistematicamente "melhores coberturas por cliques das arestas de G" é, certamente, uma prioridade a investigar no futuro.

Para o algoritmo Non Delayed Relax-and-Cut, investigamos a dualização dinâmica de duas famílias de desigualdades válidas. Respectivamente, desigualdades de clique e de buraco ímpar. Nos experimentos computacionais que efetuamos, a dualização de desigualdades de clique se mostrou muito mais efetiva que a de buracos ímpares. Vale no entanto ressaltar que o tempo disponível para trabalharmos com as mesmas foi relativamente curto. Assim sendo, é possível que não tenhamos encontrado uma forma minimamente vantajosa de utilizá-las em um ambiente Relax-and-Cut. Investigar essa questão mais a fundo é algo que pretendemos fazer futuramente.

Em particular, um dos algoritmos NDRC que implementamos, combinando nossa

reformulação do problema e a dualização dinâmica das desigualdades de cliques, se mostrou capaz de resolver (com garantia de otimalidade) diversas das instâncias de teste que consideramos. Notadamente, conseguiu obter 34 certificados de otimalidade, no universo das 119 instâncias de teste consideradas. Da mesma forma, obteve uma diferença percentual média entre seus limites superiores e inferiores da ordem de 78%, para as 119 instâncias.

A exemplo de trabalhos anteriores que utilizaram algoritmos Relax-and-Cut para inicializar de forma avançada um algoritmo Branch-and-Cut, também implementamos um algoritmo híbrido, o NDRC-CPLEX (que se utiliza do software de Programação Inteira CPLEX). Ao fazer isso, conseguimos aumentar para 64 o número de instâncias com certificados de otimalidade, aqui incluídas três instancias de teste que se encontravam abertas há mais de trinta anos. Estas são as seguintes:

- **C125.9** com valor ótimo de 34, resolvida em uma combinação de 12 segundos do NDRC com menos de 1 segundo do CPLEX.
- **MANN_a81** com valor ótimo de 1.100, resolvida em uma combinação de 2 horas e 15 minutos do NDRC com 9 minutos do CPLEX.
- **p_hat700-2** com valor ótimo de 44, resolvida em uma combinação de 4 minutos do NDRC e 1 hora e 50 minutos do CPLEX.

Em resumo, o algoritmo NDRC-CPLEX obteve prova de otimalidade para 64 das 119 instâncias de teste consideradas. Por outro lado, valores ótimos foram obtidos através de limites inferiores ou superiores, para 110 dessas instâncias. Podemos então concluir que a metodologia de solução aqui proposta é bastante atraente para o Problema do Conjunto Independente Máximo.

Vale ressaltar, no entanto, que para alguns grupos de instâncias de teste, o algoritmo NDRC teve um desempenho aquém do esperado. As razões para tanto se alternaram entre os limites inferiores e os limites superiores obtidos em cada caso. Certamente temos espaço para melhorar nossa heurística primal. Esta ficou, em diversas ocasiões, aquém das heurísticas disponíveis na literatura. Diferentes métodos de busca locais, como VNS (*Variable Neighborhood Search*), devem ser investigados futuramente afim de tentar aprimorar as soluções primais do problema. Para melhorar os limites duais, temos pelo menos dois caminhos a seguir. O primeiro é refinar o procedimento que utilizamos para encontrar coberturas por cliques das arestas de G. O segundo é encontrar formas mais efetivas de utilização das desigualdades de buraco ímpar.

Para o algoritmo exato, uma alternativa de melhora seria substituir o CPLEX por uma algoritmo Branch-and-Cut (projetado especificamente para resolver o Problema do Conjunto Dominante Máximo). Tal algoritmo, além de se beneficiar das

desigualdades de clique e, eventualmente das de buraco ímpar, transferidas diretamente pelo algoritmo NDRC, separaria estas mesmas famílias de desigualdades ao longo de sua execução.

Por fim, este trabalho se concentrou apenas no Problema do Conjunto Independente Máximo. Contudo, os algoritmos NDRC e NDRC-CPLEX que aqui desenvolvemos são aplicáveis quase sem adaptações ao Problema do Máximo Empacotamento de Conjuntos [51–53]. Na realidade, toda a metodologia de solução que aqui empregamos pode ser adaptada para aquele problema. Note que até mesmo reformular o problema através de uma cobertura por cliques das arestas de seu grafo de conflito, é algo extremamente interessante e original a fazer. Mesmo que venhamos a concluir que trabalhar diretamente com esta reformulação não seja tão interessante como antecipamos, utilizar as desigualdades de clique que a definem certamente nos ajudaria a fortalecer a formulação original do problema, a um custo computacional marginal.

Referências Bibliográficas

[1] GAVRIL, F. "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph", *SIAM Journal on Computing*, v. 1, n. 2, pp. 180–187, 1972.

[2] TARJAN, R. E., TROJANOWSKI, A. E. "Finding a maximum independent set", *SIAM Journal on Computing*, v. 6, n. 3, pp. 537–546, 1977.

[3] BACK, T., KHURI, S. "An evolutionary heuristic for the maximum independent set problem". In: *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 531–535. IEEE, 1994.

[4] GAREY, M. R., JOHNSON, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA, W. H. Freeman & Co., 1974.

[5] BALAS, E., YU, C. S. "Finding a maximum clique in an arbitrary graph", *SIAM Journal on Computing*, v. 15, n. 4, pp. 1054–1068, 1986.

[6] CARRAGHAN, R., PARDALOS, P. M. "An exact algorithm for the maximum clique problem", *Operations Research Letters*, v. 9, n. 6, pp. 375–382, 1990.

[7] BOMZE, I., BUDINICH, M., PARDALOS, P., et al. "The Maximum Clique Problem". In: Du, D. Z., Pardalos, P. (Eds.), *Handbook of Combinatorial Optimization*, Springer, pp. 1–74, Boston, MA, USA, 1999.

[8] LUCENA, A. "Non delayed relax-and-cut algorithms", *Annals of Operations Research*, v. 140, n. 1, pp. 375–410, 2005.

[9] IBM CORPORATION, I. "Introducing IBM ILOG CPLEX Optimization Studio V12.7.1". 2017. Disponível em: <`https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_relnotes_intro.html`>.

[10] PADBERG, M., RINALDI, G. "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", *SIAM Review*, v. 33, n. 1, pp. 60–100, 1991.

[11] FRIDEN, C., HERTZ, A., DE WERRA, D. "TABARIS: an exact algorithm based on tabu search for finding a maximum independent set in a graph", *Computers & Operations Research*, v. 17, n. 5, pp. 437–445, 1990.

[12] BATTITI, R., PROTASI, M. "Reactive local search for the maximum clique problem 1", *Algorithmica*, v. 29, n. 4, pp. 610–637, 2001.

[13] ANDRADE, D. V., RESENDE, M. G., WERNECK, R. F. "Fast local search for the maximum independent set problem", *Journal of Heuristics*, v. 18, n. 4, pp. 525–547, 2012.

[14] REBENNACK, S., REINELT, G., PARDALOS, P. M. "A turotial on branch and cut algorithms for the maximum stable set problem", *International Transactions in Operational Research*, v. 19, n. 1–2, pp. 161–199, 2012.

[15] NEMHAUSER, G., SIGISMONDI, G. "A Strong Cutting Plane/Branch-and-Bound Algorithm for Node Packing", *Journal of the Operational Research Society*, v. 43, n. 5, pp. 443–457, 1992.

[16] YANNAKAKIS, M. "Node-and edge-deletion NP-complete problems". In: *Proceedings of the tenth annual ACM symposium on Theory of computing*, pp. 253–264. ACM, 1978.

[17] REBENNACK, S., OSWALD, M., THEIS, D. O., et al. "A branch and cut solver for the maximum stable set problem", *Journal of Combinatorial Optimization*, v. 21, n. 4, pp. 434–457, 2011.

[18] MANNINO, C., SASSANO, A. "An exact algorithm for the maximum stable set problem", *Computational Optimization and Applications*, v. 3, n. 3, pp. 243–258, 1994.

[19] MANNINO, C., SASSANO, A. "Edge projection and the maximum cardinality stable set problem", *DIMACS series in discrete mathematics and theoretical computer science*, v. 26, pp. 205–219, 1996.

[20] BUTENKO, S., PARDALOS, P. M. *Maximum independent set and related problems, with applications*. Tese de Doutorado, University of Florida, 2003.

[21] WARRIER, D., WILHELM, W. E., WARREN, J. S., et al. "A branch-and-price approach for the maximum weight independent set problem", *Networks: An International Journal*, v. 46, n. 4, pp. 198–209, 2005.

[22] AVENALI, A. "Resolution branch and bound and an application: the maximum weighted stable set problem", *Operations research*, v. 55, n. 5, pp. 932–948, 2007.

[23] REBENNACK, S. "Maximum stable set problem: a branch & cut solver", *Ruprecht-Karls-Universitt Heidelberg, Fakultt fr Mathematik und Informatik*, 2006.

[24] CAMPELO, M., CORREA, R. C. "A Lagrangian relaxation for the maximum stable set problem", *arXiv preprint arXiv:0903.1407*, 2009.

[25] JIAN, T. "An O (2 0.304 n) algorithm for solving maximum independent set problem", *IEEE Transactions on Computers*, v. 100, n. 9, pp. 847–851, 1986.

[26] ROBSON, J. M. *Finding a maximum independent set in time O (2n/4).* Relatório técnico, Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[27] XIAO, M., NAGAMOCHI, H. "Exact algorithms for maximum independent set", *Information and Computation*, v. 255, pp. 126–146, 2017.

[28] NEMHAUSER, G. L., WOLSEY, L. A. "Integer programming and combinatorial optimization", *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, v. 20, pp. 8–12, 1988.

[29] GRÖTSCHEL, M., LOVASZ, L., SCHRIJVER, A. "The ellipsoid method and its consequences in combinatorial optimization", *Combinatorica*, v. 1, pp. 169–197, 1981.

[30] ORLIN, J. "Contentment in graph theory: Covering graphs with cliques", *Indagationes Mathematicae (Proceedings)*, v. 80, n. 5, pp. 406 – 424, 1977.

[31] KELLERMAN, E. "Determination of keyword conflict", *IBM Technical Disclosure Bulletin*, v. 16, n. 2, pp. 544–546, 1973.

[32] KOU, L. T., STOCKMEYER, L. J., WONG, C.-K. "Covering edges by cliques with regard to keyword conflicts and intersection graphs", *Communications of the ACM*, v. 21, n. 2, pp. 135–139, 1978.

[33] GRAMM, J., GUO, J., HÜFFNER, F., et al. "Data reduction, exact, and heuristic algorithms for clique cover". In: *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pp. 86–94. Society for Industrial and Applied Mathematics, 2006.

[34] LUCENA, A. "Steiner Problem in Graphs: Lagrangean Relaxation and Cutting Planes", *COAL Newsletter, Mathematical Optimization Society*, v. 21, pp. 2–7, 1992.

[35] ESCUDERO, L., GUIGNARD, M., MALIK, K. "A Lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships", *Annals of Operations Research*, v. 50, n. 1, pp. 219–237, 1994.

[36] ABOUDI, R., HALLEFJORD, A., JORNSTEN, K. "A facet generation and relaxation technique applied to an assignment problem with side constraints", *European Journal of Operational Research*, v. 3, n. 50, pp. 335–344, 1991.

[37] CALHEIROS, F. C., LUCENA, A., DE SOUZA, C. C. "Optimal rectangular partitions", *Networks*, v. 41, n. 1, pp. 51–67, 2003.

[38] MARTINHON, C. A. J., LUCENA, A., MACULAN, N. "Stronger K-tree relaxations for the vehicle routing problem", *European Journal of Operational Research*, v. 158, n. 1, pp. 56–71, 2004.

[39] DA CUNHA, A. S., LUCENA, A. "Lower and upper bounds for the degree-constrained minimum spanning tree problem", *Networks*, v. 50, n. 1, pp. 55–66, 2007.

[40] CAVALCANTE, V. F., DE SOUZA, C. C., LUCENA, A. "A Relax-and-Cut algorithm for the set partitioning problem", *Computers & OR*, v. 35, n. 6, pp. 1963–1981, 2008.

[41] DA CUNHA, A. S., LUCENA, A., MACULAN, N., et al. "A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs", *Discrete Applied Mathematics*, v. 157, n. 6, pp. 1198–1217, 2009.

[42] DA CUNHA, A. S., BAHIENSE, L., LUCENA, A., et al. "A New Lagrangian Based Branch and Bound Algorithm for the 0-1 Knapsack Problem", *Electronic Notes in Discrete Mathematics*, v. 36, pp. 623–630, 2010.

[43] ÁLVAREZ-MIRANDA, E., SINNL, M. "A Relax-and-Cut framework for large-scale maximum weight connected subgraph problems", *Computers & OR*, v. 87, pp. 63–82, 2017.

[44] FISCHETTI, M., SALVAGNIN, D. "A relax-and-cut framework for Gomory mixed-integer cuts", *Math. Program. Comput.*, v. 3, n. 2, pp. 79–102, 2011.

[45] BEASLEY, J. E. "Lagrangian Relaxation". In: Reeves, C. R. (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, John Wiley & Sons, Inc., pp. 243–303, New York, NY, USA, 1993.

[46] BELLONI, A., SAGASTIZÁBAL, C. A. "Dynamic bundle methods", *Matematical Programming*, v. 120, n. 2, pp. 289–311, 2009.

[47] MARTELLO, S. "Knapsack problems: algorithms and computer implementations", *Wiley-Interscience series in discrete mathematics and optimiza tion*, 1990.

[48] PISINGER, D. "An expanding-core algorithm for the exact 0–1 knapsack problem", *European Journal of Operational Research*, v. 87, n. 1, pp. 175–187, 1995.

[49] FONDZEFE, F. T. *Advanced Branching Rules for Maximum Stable Set Integer Programs.* Tese de Mestrado, University of L'Aquila, Italy, 2016.

[50] GLOVER, F., LAGUNA, M. "Tabu search: effective strategies for hard problems in analytics and computational science", *Handbook of Combinatorial Optimization,*, v. 21, pp. 3261–3362, 2013.

[51] DELORME, X., GANDIBLEUX, X., RODRIGUEZ, J. "GRASP for set packing problems", *European Journal of Operational Research*, v. 153, n. 3, pp. 564–580, 2004.

[52] GUO, Y., LIM, A., RODRIGUES, B., et al. "Using a Lagrangian heuristic for a combinatorial auction problem". In: *Tools with Artificial Intelligence, 2005. ICTAI 05. 17th IEEE International Conference on*, pp. 5–pp. IEEE, 2005.

[53] SVIRIDENKO, M., WARD, J. "Large neighborhood local search for the maximum set packing problem". In: *International Colloquium on Automata, Languages, and Programming*, pp. 792–803. Springer, 2013.