Problema de roteamento em anéis de dois níveis

Cecília Lescano Osório

Problema de roteamento em anéis de dois níveis

Cecília Lescano Osório

Orientadora: *Profª. Drª. Edna Ayako Hoshino*

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação no Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Mato Grosso do Sul.

UFMS - Campo Grande
dezembro/2021

Abstract

In this dissertation, we study the ring/ring problem, which consists in, given a graph and costs associated to its edges, designing a hierarchical network in two levels, with both levels being rings. We present an integer linear programming model for the problem and propose an exact branch-and-price algorithm to solve it. Since the proposed model uses an exponential number of variables, we use the column generation method to solve the linear relaxation. We also propose a relaxation for the column generation and heuristics to improve the performance of the proposed algorithm.

Resumo

Nesta dissertação de mestrado estudamos o problema do roteamento em anéis de dois níveis, que consiste em, dado um grafo e custos associados às arestas, projetar uma rede hierárquica em dois níveis em que ambos os níveis são anéis. Apresentamos um modelo em programação linear inteira para o problema e propomos um algoritmo exato *branch-and-price* para resolvê-lo. Uma vez que o modelo proposto faz uso de um número exponencial de variáveis, utilizamos o método da geração de colunas para resolver a relaxação linear do modelo. Propomos também uma relaxação para a geração de colunas e heurísticas para melhorar o desempenho do algoritmo proposto.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO

1

Introdução

O problema de roteamento em anéis de dois níveis (RRP), do inglês *Ring/Ring Problem*, é um problema que consiste na construção de uma rede hierárquica em dois níveis, como as redes de transportes ou as de comunicações, em que cada nível é um anel ou ciclo, ou ainda *ring*. Seguindo o que é usado na literatura, usaremos as denominações da área de telecomunicações.

A rede *ring/ring* é formada por um ciclo principal, chamado de *backbone*, e de um número de ciclos secundários, chamados redes de acesso. Cada rede de acesso possui um único *hub*, um nó especial que faz parte do *backbone* e de uma ou mais redes de acesso.

Uma rede *ring/ring* é construída a partir de uma lista de nós, na qual há um nó especial denominado depósito e identificado pelo rótulo 0, que sempre deve estar no *backbone* e não pode ser um *hub*, ou seja, não pode ter redes de acesso conectadas diretamente a ele. A Figura 1.1 ilustra um exemplo de uma rede em anéis de dois níveis. O nó 2, em destaque, é um *hub*. As arestas contínuas formam o *backbone*, e as arestas tracejadas formam as redes de acesso.

Para cada par de nós (u,v), há um custo b_{uv} para construir uma ligação conectando-os no *backbone* e um custo c_{uv}, em geral mais barato, para conectá-los em uma rede de acesso. Além disso, cada nó i possui um custo f_i para a instalação de uma facilidade nele para que possa ser um *hub* na rede. Por fim, existem dois inteiros Q e k que impõem limites na configuração da rede: o primeiro indica um limite para o número de nós em um ciclo e o segundo, um limite para o número de ciclos secundários que podem estar conectados em um mesmo *hub*.

Respeitando esses limites, o objetivo do RRP consiste em construir uma

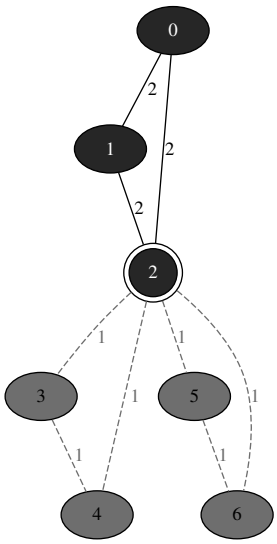

Figura 1.1: Exemplo de rede *ring/ring*.

rede *ring/ring* de custo total mínimo. O custo total de uma rede é dado pela soma dos custos das ligações dos nós na rede *backbone* mais os custos das conexões dos nós nas redes de acesso e os custos das instalações de facilidades nos nós *hubs*.

O objetivo deste trabalho é avaliar o uso do método da geração de colunas para resolver o RRP e compará-lo com outros métodos estudados na literatura. A ideia é modelar o problema por programação linear inteira e implementar um algoritmo *branch-and-price*. Para melhorar o desempenho do algoritmo, incorporamos heurísticas primais e avaliamos o uso de outras técnicas para acelerar a geração de colunas.

No Capítulo 2, apresentamos os trabalhos relacionados. A metodologia é descrita no Capítulo 3. No Capítulo 4, apresentamos o trabalho desenvolvido e, no Capítulo 5, descrevemos os experimentos computacionais realizados em instâncias da literatura e apresentamos uma análise dos resultados. Por fim, no Capítulo 6, discutimos as contribuições do trabalho.

CAPÍTULO

2

Trabalhos Relacionados

O problema da rede *ring/ring* foi estudado por Thomadsen and Stidsen (2005) na forma *ring/1-ring*, ou seja, com $k = 1$. Sua abordagem considera demandas de comunicação que devem ser atendidas da forma mais barata possível. O problema é dividido em etapas que incluem o agrupamento de nós para formar as redes de acesso e a escolha dos *hubs*, para que depois sejam escolhidas as ligações. Essa escolha simplifica o processo de otimização, mas, de acordo com os autores, pode resultar em uma solução sub-ótima. Um modelo *branch-and-price* para uma versão modificada do problema, incluindo algumas das etapas apresentadas, é introduzido e os resultados são apresentados para redes de até 36 nós.

Rodríguez-Martín et al. (2016a) apresentam uma abordagem *branch-and-cut* para o projeto de redes *ring/ring* e também para as redes *ring/2-edge-connected*, *2-edge-connected/ring* e *2-edge-connected/2-edge-connected*. Um grafo do tipo *ring* é um tipo específico de grafo *2-edge-connected*. Esses tipos de rede são estudados por terem todos a característica de robustez para a falha em uma ligação. As redes *ring/ring* estudadas também são da forma *ring/1-ring*. O vértice inicial ou depósito pode ser um *hub*, e o custo de hub é pago por todos os vértices do *backbone*, independentemente de estarem ligados a uma rede de acesso ou não. O limitante Q não se aplica ao *backbone*.

Em Rodríguez-Martín et al. (2016b) é introduzida a forma *ring/k-rings*, em que são permitidas k redes de acesso em cada *hub*. Como no trabalho anterior, é utilizada a abordagem *branch-and-cut*. Nesta definição, todos os vértices do *backbone*, incluindo o depósito, devem ser *hubs* e estar em pelo menos uma rede de acesso. O limitante Q aqui também não se aplica ao *backbone*.

Em Alumur et al. (2021), os autores analisam a modelagem de diversos

problemas de *hub location*.

Também é importante citar os vários trabalhos realizados a respeito de outros tipos de redes hierárquicas. Um exemplo são as redes do tipo *ring/tree*, que utilizam ciclos e árvores. O *Capacitated Ring Tree Problem* (CRTP), que considera demandas de dois tipos de clientes, é estudado em Hoshino and Hill (2014) e em Hill and Voß (2016). O objetivo também é minimizar os custos das arestas, o que se assemelha ao nosso trabalho.

Outro tipo de rede hierárquica muito estudado é a *ring/star*. O *Capacitated m-Ring-Star Problem* (CmRSP) consiste em projetar um conjunto de ciclos que passem por um depósito central e por pontos de transição e/ou clientes, e associar cada cliente não visitado a um ponto ou a um cliente visitado por um dos ciclos. Mais uma vez devem ser minimizados os custos. O problema foi introduzido por Baldacci et al. (2007), e uma abordagem *branch-and-cut-and-price* foi sugerida por Hoshino and De Souza (2012).

O RRP também possui características em comum com o *Vehicle Routing Problem* (VRP), introduzido por Dantzig and Ramser (1959), em que são determinadas rotas para que veículos saiam de um depósito, atendam a clientes e retornem ao depósito. O VRP também utiliza ciclos para conectar os clientes, mas não há distinção entre ciclo principal e secundários.

CAPÍTULO

3

Metodologia

Neste capítulo são elencados os principais conceitos e métodos utilizados para o entendimento do problema estudado.

3.1 Programação Linear Inteira

De acordo com Lewis (2008), podemos caracterizar um **problema de programação linear** (PL) na seguinte forma padrão:

$$\begin{array}{lr} (P) \quad \text{minimizar} & c_1x_1 + \ldots + c_nx_n = z \\ \text{sujeito a} & a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ & \vdots \\ & a_{m1}x_1 + \ldots + a_{mn}x_n = b_m \\ & x_1, \quad \ldots, \qquad x_n \geq 0. \end{array}$$

O modelo do problema (P) também pode ser representado em sua forma matricial a seguir, em que A é uma matriz $m \times n$, c é um vetor de tamanho n, chamado **vetor custo**, e b é um vetor de tamanho m:

$$\begin{array}{lr} (P) \quad \text{minimizar} & cx = z \\ \text{sujeito a} & Ax = b \\ & x \geq 0. \end{array}$$

Em um PL, a expressão z sendo otimizada é chamada **função objetivo**. As variáveis $x_1, \ldots, x_n$ são chamadas de **variáveis de decisão** e seus valores são sujeitos a m restrições, além das restrições de não-negatividade. Um conjunto de valores para $x_1, \ldots, x_n$ que satisfaça a todas as restrições é chamado de ponto

viável ou **solução viável** e o conjunto de todos os pontos viáveis é chamado de **região viável**. A **solução ótima** é um ponto da região viável que minimiza a função objetivo.

Um **problema de programação linear inteira** (PLI) é um problema de programação linear em que as variáveis são discretas, isto é, devem possuir valores inteiros.

O problema P descrito a seguir ilustra um exemplo de PLI:

$$(P) \quad \text{minimizar } cx = z \tag{3.1}$$

$$\text{sujeito a } Ax = b \tag{3.2}$$

$$x \in \mathbb{Z}^n_+. \tag{3.3}$$

A restrição (3.3) é chamada **restrição de integralidade**.

A **relaxação linear** de um problema de PLI (P) consiste no problema linear obtido de (P) pela remoção das suas restrições de integralidade. O valor da solução ótima da relaxação linear de (P) é um limitante inferior para o valor ótimo de (P).

3.2 Método Simplex

Como definido por Dantzig et al. (1955), o método Simplex é um método iterativo finito para problemas que envolvem inequações lineares, muito utilizado para a resolução de problemas de programação linear.

A explicação a seguir é baseada em Wolsey (1998).

Um **poliedro** é o conjunto de pontos do $\mathbb{R}^n$ que satisfazem um conjunto finito de equações e inequações lineares. Em otimização combinatória, o poliedro representa o conjunto de soluções viáveis de um problema. Seja $S = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ um poliedro com $A : m \times n$ e $n \geq m$.

Uma solução viável x de S é uma **solução básica** se $x_i = 0$ para todo $i = m+1, \ldots, n$ e a matriz $B = [A_{.1}, A_{.2}, \ldots, A_{.m}]$, chamada de **base**, é inversível. Considere que as colunas da matriz A são rearranjadas para que as primeiras m colunas formem a base. Aqui, a notação $A_{.j}$ é usada para representar o vetor coluna da j-ésima coluna da matriz A. Além disso, o ponto x é uma solução básica de S se, e somente se, x é um ponto extremo de S.

Sabe-se que um problema $z = min\{cx : x \in S\}$ que tem valor finito possui um ponto extremo ótimo, ou seja, um ponto extremo que é uma solução ótima.

O algoritmo Simplex parte de um ponto extremo ou solução básica inicial e, a partir dele, busca por um ponto extremo vizinho que tenha um valor melhor na função objetivo. Cada solução básica corresponde a uma base, a matriz B definida anteriormente. As m variáveis correspondentes às colunas de A que

estão em B são chamadas **variáveis básicas**, e as restantes são chamadas **variáveis não básicas**.

Em cada iteração do algoritmo Simplex encontra-se uma nova base e um novo conjunto de variáveis básicas. Para isso, o algoritmo realiza um passo chamado ***pricing***, que consiste em escolher uma variável não básica para entrar na base que possa melhorar o valor da função objetivo. Essa melhoria é chamada de **custo reduzido** ($\bar{c}_j$) e é dada por:

$$\bar{c}_j = c_j - uA_{.j}$$

O valor de u pode ser calculado como $u = c_B B^{-1}$ e c_B corresponde ao vetor custo das variáveis básicas e B^{-1} a inversa da base B.

O ponto extremo corrente é uma solução ótima quando não existe mais variável não básica de custo reduzido negativo.

3.3 Geração de Colunas

A **geração de colunas** é um método utilizado para resolver problemas de programação linear em que o número de variáveis n é exponencial em relação ao número de restrições m.

Como explicado por Lübbecke (2010), a partir de um problema original chamado **problema mestre** (PM), consideramos um outro problema chamado **problema mestre reduzido** (PMR), que possui apenas um subconjunto das variáveis do problema mestre.

O modelo a seguir representa um problema mestre com $|J|$ exponencial em $|I|$.

$$\begin{aligned} \text{v(PM):= min} \quad & \sum_{j \in J} c_j \cdot \lambda_j \\ \text{sujeito a} \quad & \sum_{j \in J} a_j \cdot \lambda_j = b, \forall i \in I \\ & \lambda_j \geq 0, j \in J. \end{aligned}$$

O PMR é apresentado no modelo a seguir em que $\tilde{J} \subset J$ e $|\tilde{J}| \ll |J|$.

$$\begin{aligned} \text{v(PMR):= min} \quad & \sum_{j \in \tilde{J}} c_j \cdot \lambda_j \\ \text{sujeito a} \quad & \sum_{j \in \tilde{J}} a_j \cdot \lambda_j = b, \forall i \in I \\ & \lambda_j \geq 0, j \in \tilde{J}. \end{aligned}$$

Ao aplicar o método Simplex, buscamos no passo de *pricing* uma variável não-básica de custo reduzido negativo para ser incluída em $\tilde{J}$. Uma vez que a quantidade de varíaveis não-básicas em $J \setminus \tilde{J}$ é exponencial em $|I|$, torna-se

impraticável calcular explicitamente o custo reduzido de cada variável não-básica. Para resolver este problema, o método da geração de colunas consiste em encontrar uma variável não-básica de custo reduzido negativo, se houver, resolvendo-se um outro problema de otimização, chamado **problema de *pricing*** (PP), descrito a seguir:

$$v(PP) := min\{c_j - \pi^* a_j | j \in J\}.$$

Na equação acima, π^* representa uma solução dual ótima do PMR.

Quando o valor da solução ótima do problema de *pricing* é negativo, isto é, $v(PP) < 0$, a variável λ_j e sua coluna de coeficientes (c_j, a_j) são adicionadas ao problema mestre reduzido. O processo é iterado até que não sejam encontradas mais variáveis de custo reduzido negativo. Nessas condições, é garantido que a solução ótima do problema mestre reduzido também é a solução ótima do problema mestre.

Ainda de acordo com Lübbecke (2010), a geração de colunas é usada em problemas com um número exponencial de variáveis, em especial, naqueles em que o conjunto de variáveis pode ser descrito como a região viável de outro problema de otimização, fazendo com que não seja necessário listar as variáveis explicitamente para a busca daquela que possui o melhor custo reduzido. Além disso, em várias aplicações este problema de otimização (problema de *pricing*) também é um problema de programação linear.

3.4 Branch-and-bound

Branch-and-bound é uma estratégia comumente utilizada para solucionar problemas de PLI que se assemelha a algoritmos de divisão e conquista pois, assim como eles, consiste na partição do problema original. No entanto, enquanto a divisão e conquista particiona uma instância em instâncias menores, o *branch-and-bound* particiona o espaço de soluções em espaços menores, o que equivale a dividir o problema em outros problemas diferentes e potencialmente mais fáceis.

Como explicado por Wolsey (1998), o *branch-and-bound* decompõe o problema inicial em subproblemas usando uma estrutura de árvore, chamada árvore de *branch-and-bound*. A ideia geral é dividir o espaço de soluções S de um problema $z = min\{f(x) : x \in S\}$ em S^1 e S^2 tais que $S = S^1 \cup S^2$, definindo subproblemas $z^k = min\{f(x) : x \in S^k\}$, para $k = 1,2$. De fato, pode-se particionar S em vários subespaços $S^1, S^2, \ldots, S^r$ desde que $S = \bigcup_{k=1,\ldots,r} S^k$. Desta forma, o valor de z do problema original pode ser calculado por $z = min_{k=1,\ldots,r} z^k$. O processo de se dividir um problema em subproblemas é chamado de **ramificação**, ou ***branching***, e cada subproblema é representado como um nó filho do nó que

foi particionado, na árvore de *branch-and-bound*.

A explosão combinatória da árvore de *branch-and-bound* é evitada pelo algoritmo por meio de uma operação de poda da árvore e com a utilização de dois limitantes chamados **limitante primal** e **limitante dual**. O limitante primal, em um problema de minimização, consiste em um valor ub que define um limite superior para z, ou seja, $z \leq ub$. Por outro lado, um limitante dual consiste em um valor lb que define um limite inferior para z, ou seja, $z \geq lb$. Em geral, heurísticas, chamada **heurísticas primais**, são usadas para obter os limitantes primais e relaxações são usadas para obter os limitantes duais. O algoritmo *branch-and-bound* calcula um limitante inferior $lb(P_i)$ para o problema P_i associado ao nó i da árvore. Desta forma, um nó i é podado da árvore se $lb(P_i) \geq ub$. Há outras duas formas de podas, por inviabilidade e por otimalidade. Na poda por inviabilidade, um nó é podado se o espaço de soluções do problema associado ao nó é vazio. Por fim, um nó é podado por otimalidade se $lb(P_i) = ub(P_i)$. Usualmente, o valor de $lb(P_i)$ é obtido pela relaxação linear do problema P_i e se a solução ótima da relaxação linear satisfaz as restrições de integralidade, temos que $lb(P_i) = ub(P_i)$ e o problema é podado por otimalidade.

3.5 Branch-and-price

Como definido por Wolsey (1998), o *branch-and-price* pode ser descrito como uma combinação do *branch-and-bound* e do método da geração de colunas. Barnhart et al. (1970) explicam que o *branch-and-price* deixa de fora colunas do PL e utiliza o problema de *pricing* para checar a otimalidade e buscar colunas para entrar na base. Quando nenhuma coluna é encontrada, realiza-se uma fase de *branching* para particionar o problema em subproblemas. É importante observar que o método da geração de colunas é usado para resolver a relaxação linear de cada subproblema, associado a nós da árvore de *branch-and-bound*. Portanto, o método da geração de colunas é aplicado sucessivas vezes em cada nó, até que nenhuma variável de custo reduzido negativo exista, e também é usado diversas vezes, um em cada nó da árvore. Desta forma, o desempenho de um algoritmo *branch-and-price* depende da formulação de PLI e do problema de *pricing*, além das heurísticas primais.

3.6 Restrições de Eliminação de Subciclos Miller-Tucker-Zemlin (MTZ)

Nossa formulação para o RRP utiliza as restrições de eliminação de subciclos introduzidas em Miller et al. (1960) para uma variante do problema do caixeiro viajante, conhecidas como MTZ. Nesse problema, um “caixeiro” deve

partir de uma cidade inicial 0 e visitar n cidades, voltando a 0 exatamente t vezes. Cada sequência de cidades visitadas entre as visitas a 0 é chamada de ciclo e cada ciclo não deve ter mais do que p cidades. Observe que uma solução desse problema consiste em t ciclos, cada um deles contendo o vértice 0 e de comprimento no máximo p.

As restrições de eliminação de subciclos têm como objetivo eliminar qualquer ciclo que tenha mais do que p cidades ou que não comecem e terminem na cidade 0. Para isso, atribui-se a cada cidade i do ciclo um rótulo u_i. Para toda aresta (i, j) do ciclo, obriga-se que $u_j > u_i$, com exceção de quando $j = 0$.

Considere que x_{ij} é uma variável de decisão associada a cada aresta do grafo e tem valor igual a 1 se e somente se a aresta $(i, j,)$ é selecionada para formar o ciclo. As restrições de MTZ são as seguintes:

$$u_i - u_j + p \cdot x_{ij} \leq p - 1, (1 \leq i \neq j \leq n)$$

O valor de cada rótulo u_i pode ser um número real arbitrário, mas em geral, obriga-se que o seu valor seja um inteiro no intervalo $[1, n]$.

A Figura 3.1 mostra um grafo que não pode ser gerado em virtude das restrições MTZ. Observe que as arestas (c, a) e (f, d) ligam um vértice a outro com rótulo menor. As restrições MTZ permitem que apenas uma aresta desse tipo seja utilizada. De fato, apenas aquela que fecha o ciclo, ou seja, a que liga um vértice do ciclo ao vértice inicial 0.

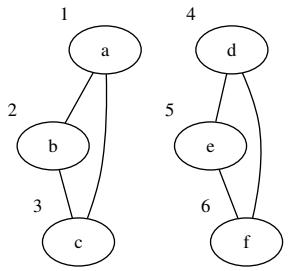

Figura 3.1: Exemplo de grafo que não pode ser formado devido às restrições MTZ.

A Figura 3.2 mostra outro exemplo, em que as arestas (c, a) e (e, c) não podem estar ambas no grafo.

3.7 Relaxação ng-route

A relaxação *ng-route* para o problema do roteamento capacitado de veículos, proposta em Baldacci et al. (2011), gera uma rota em um grafo começando e terminando no mesmo vértice s, respeitando uma dada capacidade Q e permitindo repetição de vértices em certas condições.

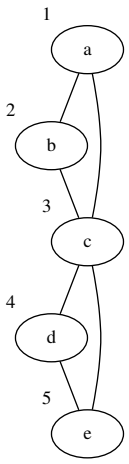

Figura 3.2: Outro exemplo de grafo que não pode ser formado devido às restrições MTZ.

O algoritmo recebe um inteiro Δ e um conjunto $NG(i) \subseteq V$ para cada vértice i do grafo, tais que $i \in NG(i)$ e $|NG(i)| \leq \Delta$. São gerados caminhos parciais chamados *ng-paths*, começando em um dado vértice s, nos quais a repetição de um vértice i só é permitida se entre as duas ocorrências de i existir pelo menos um vértice j tal que $i \notin NG(j)$. Caminhos viáveis, que não superaram a capacidade, são estendidos para construir *ng-routes*. Além disso, cada vértice i do grafo é associado a uma demanda d_i de modo que a soma das demandas de todos os vértices do *ng-route* não deve ultrapassar a capacidade Q.

Por exemplo, considere um grafo G com vértices i, j e k e suponha que $NG(i) = \{a, i\}$, $NG(j) = \{b, j\}$ e $NG(k) = \{k, i\}$. A Figura 3.3 ilustra um exemplo de caminho que pode ser estendido com a inclusão do vértice i, enquanto a Figura 3.4 apresenta um caminho em que a inclusão do vértice i não é permitida.

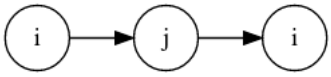

Figura 3.3: O vértice i pode ser incluído ao caminho porque $i \notin NG(j)$.

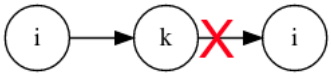

Figura 3.4: O vértice i não pode ser incluído ao caminho porque $i \in NG(k)$.

Um *ng-path* começando em s é um caminho $p = (s = v_0, v_1, \ldots, v_l)$ tal que se $v_i = v_j$, então existe um r, $i < r < j$ tal que $v_i \notin NG(v_r)$. Seja $\Pi(p) = \{v_r : v_r \in \bigcap_{t=r+1}^{l} NG(v_t), r = 1, \ldots, l-1\} \cup \{v_l\}$. Os *ng-paths* $p = (s, v_1, \ldots, v_l)$ são estendidos com a inclusão de uma aresta (v_l, v_m) apenas se $v_m \notin \Pi(p)$. O algoritmo começa com *ng-paths* (s, v) para cada vértice $v \neq s$, e termina quando a capacidade Q é atingida.

A implementação de um algoritmo para encontrar um *ng-route* de custo mínimo é feita por programação dinâmica, na qual armazena-se em cada entrada de $f(j,q)$, todos os *ng-path*s entre o vértice inicial s e j com capacidade q.

Regras de dominância são aplicadas para eliminar caminhos (*ng-paths*) que são desnecessários para gerar o melhor *ng*-route.

3.8 Metaheurística GRASP

A metaheurística GRASP (*Greedy Randomized Adaptive Search Procedure*) foi inicialmente apresentada por Feo and Resende (1989). Trata-se de um algoritmo com duas fases: uma construtiva e uma de busca local. Ambas as fases são executadas *nruns* vezes e as melhores *nsols* soluções geradas são armazenadas e devolvidas pelo algoritmo.

Na primeira fase, o algoritmo constrói uma solução de forma iterativa, incluindo um elemento por vez na solução. Diferentemente dos algoritmos gulosos que escolhem o elemento a ser incluído na solução usando um critério puramente guloso, o GRASP utiliza um parâmetro α que define um **critério de aleatoriedade** para construir uma **lista de candidatos restrita (RCL)**. Em seguida, seleciona de forma pseudoaleatória um elemento da RCL para ser incluído na solução. O parâmetro α é um valor real no intervalo $[0,1]$.

Seja C o conjunto de todos os elementos candidatos a serem incluídos na solução e $f : C \mapsto \mathbb{R}_+$ uma função, chamada **função gulosa adaptativa**, que estima o ganho $f(i)$ na função objetivo do problema ao incluir o elemento i na solução. Considere $f_{min} = \min_{i \in C} f(i)$ e $f_{max} = \max_{i \in C} f(i)$. A RCL é construída usando-se a seguinte fórmula:

$$RCL = \{i \in C : \alpha \cdot f_{min} \leq f(i) \leq (1-\alpha) \cdot f_{max}\}.$$

Note que $\alpha = 1$ define um critério puramente guloso enquanto $\alpha = 0$ define um critério puramente aleatório. A escolha do parâmetro α afeta a qualidade da solução gerada pelo GRASP.

A segunda fase consiste em aplicar algoritmos de busca local para melhorar a solução gerada na primeira fase. O Algoritmo 1 apresenta o pseudocódigo do GRASP.

A primeira fase do algoritmo é realizada pela função `ConstroiSolucao`, cujo pseudocódigo está apresentado no Algoritmo 2. Após aplicação da fase de busca local, a solução construída é usada para atualizar o conjunto *Solucoes* com as *nsols* melhores soluções, realizada pela função `AtualizaMelhores`.

A técnica pode ser adaptada para diferentes problemas de otimização trocando-se a função gulosa adaptativa f, atualizando a lista de candidatos apropriedamente e usando uma busca local específica ao problema.

Algoritmo 1: GRASP

Data: $\alpha, nsols, nruns$
Result: *Solucoes*

```
1  Solucoes ← ∅;
2  total = 0;
3  for i = 1 até nruns do
4  |  S = ConstroiSolucao(α);
5  |  S' = BuscaLocal(S);
6  |  AtualizaMelhores(Solucoes, S', total, nsols);
7  end
```

Algoritmo 2: ConstroiSolucao

Data: α
Result: *solucao*

```
1  solucao ← ∅;
2  Constroi lista de candidatos C;
3  Calcule f[i], ∀i ∈ C;
4  f_min ← min{f[i] : i ∈ C};
5  f_max ← max{f[i] : i ∈ C};
6  while solucao não é válida e C ≠ ∅ do
7  |  LCR = {i ∈ C : α f_min ≤ f[i] ≤ (1 − α) * f_max};
8  |  Sorteia r da LCR;
9  |  solucao ← solucao ∪ {r};
10 |  Atualiza lista de candidato C;
11 |  Atualiza f, f_min e f_max;
12 end
13 if solucao é válida then
14 |  Devolva solucao;
15 end
16 Devolva ∅;
```

CAPÍTULO

4

Trabalho Desenvolvido

Neste capítulo apresentamos as formulações por PLI que propusemos para modelar o RRP e os algoritmos desenvolvidos para resolver o problema. Inicialmente, uma descrição formal do problema é apresentada e ao final do capítulo são discutidas as melhorias incorporadas aos algoritmos para acelerar o seu desempenho.

4.1 Definição Formal do Problema

Seja $G = (V,E)$ um grafo completo não-orientado com um nó especial $d \in V$ representando um depósito. Sejam $s : E \mapsto \mathbb{Z}_+$ e $b : E \mapsto \mathbb{Z}_+$ funções de custo das arestas, $f : V \mapsto \mathbb{N}$ uma função de custo de instalação de facilidade em um vértice, e dois inteiros k e Q.

Um **anel** ou **ciclo** C em G é uma sequência de vértices $(v_0, v_1, \ldots, v_l = v_0)$ tais que $\{v_i, v_{i+1}\} \in E$, $0 \leq i < l$ e $v_i \neq v_j, \forall i \neq j$. Denotamos por $E(C)$ o subconjunto de arestas $\{\{v_i, v_{i+1}\} : i = 0, \ldots, l-1\}$ e por $V(C)$ o subconjunto $\{v_i : 0 \leq i < l\}$. O tamanho, ou comprimento, de C é dado pelo inteiro $l = |E(C)|$.

Uma **rede de anéis em dois níveis**, ou ainda uma **rede *ring/ring*** de G é um par ordenado $T = (B,R)$ tal que:

(i) B é um ciclo em G que contém o vértice d;

(ii) R é um conjunto de ciclos em G;

(iii) $|V(r) \cap V(B)| = 1$, para cada ciclo $r \in R$;

(iv) $V(r) \cap V(r') = \varnothing$ ou $V(r) \cap V(r') = V(r) \cap V(B) = V(r') \cap V(B)$, para ciclos $r, r' \in R$, $r \neq r'$.

Usando a nomenclatura da área de telecomunicações, como mencionado no Capítulo 1, os vértices em $V(B) \cap V(R)$ são chamados de *hubs*, o ciclo B é a rede de *backbone* e cada ciclo em R define uma rede de acesso.

O **total de redes de acesso em um *hub*** i é denotado por $n(i)$ e é calculado por $n(i) = |\{r \in R : V(r) \cap V(B) = i\}|$.

O **custo de uma rede *ring/ring*** $T = (B,R)$ é calculado pela seguinte equação:

$$c(T) = \sum_{e \in E(B)} b(e) + \sum_{e \in E(R)} s(e) + \sum_{i \in V(B) \cap V(R)} f(i). \tag{4.1}$$

Definição 4.1 (def1) *Dado um grafo $G = (V,E)$ com custos $b : E \mapsto \mathbb{Z}_+$, $s : E \mapsto \mathbb{Z}_+$, $f : V \mapsto \mathbb{N}$, inteiros Q e k, o* **problema de roteamento em anéis de dois níveis** *consiste em encontrar uma rede ring/ring $T = (B,R)$ em G tal que:*

(i) $V = V(B) \cup V(R)$*;*

(ii) $|V(B)| \leq Q$ *e* $|V(r)| \leq Q, \forall r \in R$*;*

(iii) $n(i) \leq k$*, para todo* $i \in V(B) \setminus \{d\} \cap V(R)$*,* $n(d) = 0$*; e*

(iv) $c(T)$ *é mínimo.*

Rodríguez-Martín et al. (2016a) e Rodríguez-Martín et al. (2016b) definiram o RRP de forma um pouco diferente da que descrevemos e consideramos neste trabalho. Essas diferenças tornam os resultados reportados nesta dissertação incomparáveis com os da literatura. Diante disso, optamos por, adicionalmente, também considerar as duas definições usadas por Rodríguez-Martín et al. (2016a) e por Rodríguez-Martín et al. (2016b), as quais são descritas em seguida.

Definição 4.2 (def2) *Dado um grafo $G = (V,E)$ com custos $b : E \mapsto \mathbb{Z}_+$, $s : E \mapsto \mathbb{Z}_+$, $f : V \mapsto \mathbb{N}$, inteiros Q e k, o* **problema de roteamento em anéis de dois níveis** *consiste em encontrar uma rede ring/ring $T = (B,R)$ em G tal que:*

(i) $V = V(B) \cup V(R)$*;*

(ii) $|V(r)| \leq Q, \forall r \in R$*;*

(iii) $n(i) \leq k$*, para todo* $i \in V(B) \cap V(R)$*; e*

(iv) $\sum_{e \in E(B)} b(e) + \sum_{e \in E(R)} s(e) + \sum_{i \in V(B)} f(i)$ *é mínimo.*

Definição 4.3 (def3) *Dado um grafo $G = (V,E)$ com custos $b : E \mapsto \mathbb{Z}_+$, $s : E \mapsto \mathbb{Z}_+$, $f : V \mapsto \mathbb{N}$, inteiros Q e k, o* **problema de roteamento em anéis de dois níveis** *consiste em encontrar uma rede ring/ring $T = (B,R)$ em G tal que:*

(i) $V = V(B) \cup V(R)$;

(ii) $|V(r)| \leq Q, \forall r \in R$;

(iii) $1 \leq n(i) \leq k$, *para todo* $i \in V(B) \cap V(R)$;

(iv) $\sum_{e \in E(B)} b(e) + \sum_{e \in E(R)} s(e) + \sum_{i \in V(B)} f(i)$ *é mínimo.*

A Definição 4.2 é considerada por Rodríguez-Martín et al. (2016a) e difere da definição adotada neste trabalho em três aspectos: (1) a capacidade Q é imposta apenas às rede de acesso; (2) permite que redes de acesso usem o depósito como um *hub*; e (3) o custo de instalação de facilidade é considerado a todo vértice da rede de *backbone*, independentemente dele ser um *hub* ou não.

Já a Definição 4.3 que é considerada em Rodríguez-Martín et al. (2016b), difere da Definição 4.2 ao impor que todo vértice do *backbone* tenha pelo menos uma rede de acesso, ou seja $n(i) \geq 1$, para todo vértice $i \in V(B)$.

As Figuras 4.1, 4.2 e 4.3 ilustram um exemplo de redes construídas usando cada definição a partir dos mesmos dados de entrada.

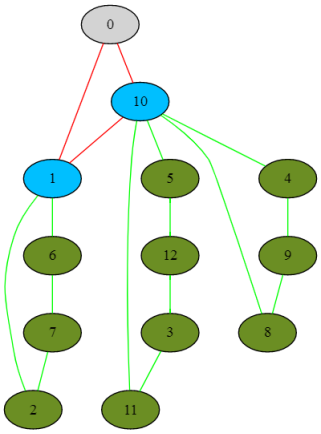

Figura 4.1: Exemplo de rede *ring/ring* com 13 vértices, $k = 3$ e $Q = 5$, construída usando a Definição 4.1. Valor da solução = 144,0.

4.2 Modelo Estendido para o RRP

Seja $\mathcal{P}$ o conjunto de todos os ciclos em G com tamanho menor ou igual a Q. Consideramos uma partição de $\mathcal{P}$ em $\mathcal{P}_1$ e $\mathcal{P}_2$, sendo que $\mathcal{P}_1$ contém todos os ciclos que passam pelo depósito e representam os possíveis *backbone*s enquanto $\mathcal{P}_2$ denota o conjunto de todos os possíveis ciclos secundários.

Nosso modelo seleciona ciclos em $\mathcal{P}$ para construir uma rede *ring/ring* de custo mínimo. Para isso, utilizamos a variável de decisão λ_p, que é igual a 1 se o ciclo p é escolhido.

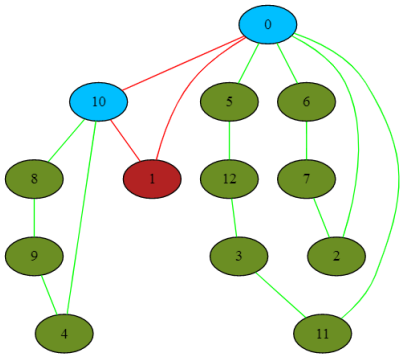

Figura 4.2: Exemplo de rede *ring/ring* com 13 vértices, $k = 3$ e $Q = 5$, construída usando a Definição 4.2. Valor da solução = 142,0.

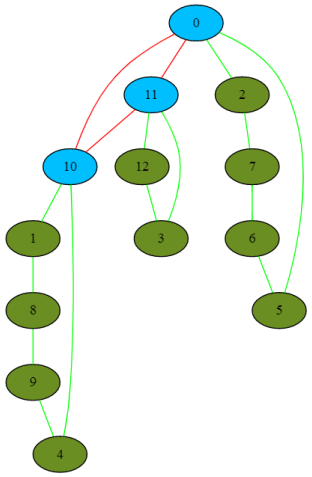

Figura 4.3: Exemplo de rede *ring/ring* com 13 vértices, $k = 3$ e $Q = 5$, construída usando a Definição 4.3. Valor da solução = 148,0.

Além de λ_p, usamos também as seguintes variáveis:

$$w_i = \begin{cases} 1, & \text{se o nó } i \text{ é selecionado para ter facilidades instaladas} \\ 0, & \text{caso contrário.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{se vértice } i \text{ pertence ao } \textit{backbone} \\ 0, & \text{caso contrário.} \end{cases}$$

$$x_e = \begin{cases} 1, & \text{se aresta } e \text{ pertence à solução} \\ 0, & \text{caso contrário.} \end{cases}$$

Para cada ciclo p em $\mathcal{P}$, considere a^p o vetor característico de dimensão $|V|+|E|$, no qual $a_i^p = 1$ ($a_e^p = 1$) se e somente se o vértice i (aresta e) pertence a $V(p)$ ($E(p)$) e $h_i^p = 1$ se e somente se o nó i em p é o *hub* de p.

O custo de um ciclo $p \in \mathcal{P}_1$, considerando-se apenas os custos de conectar os vértices do ciclo em uma rede de *backbone*, é denotado por b_p e é dado pela

seguinte equação:

$$b_p = \sum_{\{i,j\} \in E} b_{\{i,j\}} a^p_{\{i,j\}}. \tag{4.2}$$

Já o custo de conectar os vértices de um ciclo $p \in \mathcal{P}_2$ em uma rede de acesso é denotado por c_p e é dado pela seguinte equação:

$$c_p = \sum_{\{i,j\} \in E} s_{\{i,j\}} a^p_{\{i,j\}}. \tag{4.3}$$

O modelo para o RRP é como segue:

$$(F1)\ \min \sum_{p \in \mathcal{P}_1} b_p \lambda_p + \sum_{p \in \mathcal{P}_2} c_p \lambda_p + \sum_{i \in V \setminus \{d\}} f_i w_i \tag{4.4}$$

$$s.a. \sum_{p \in \mathcal{P}_1} \lambda_p = 1 \qquad (\pi) \tag{4.5}$$

$$\sum_{p \in \mathcal{P}_1} a^p_i \lambda_p = y_i \qquad \forall i \in V \setminus \{d\} \quad (\alpha_i) \tag{4.6}$$

$$\sum_{p \in \mathcal{P}_2} a^p_i \lambda_p \geq 1 - y_i \qquad \forall i \in V \setminus \{d\} \quad (\beta_i) \tag{4.7}$$

$$\sum_{p \in \mathcal{P}_2} h^p_i \lambda_p \leq k w_i \qquad \forall i \in V \setminus \{d\} \quad (\theta_i) \tag{4.8}$$

$$\sum_{p \in \mathcal{P}_1 + \mathcal{P}_2} \sum_{i \in V} a^p_i \lambda_p = n + \sum_{p \in \mathcal{P}_2} \lambda_p \qquad (\gamma) \tag{4.9}$$

$$\sum_{p \in \mathcal{P}_1 + \mathcal{P}_2} a^p_e \lambda_p = x_e \qquad \forall e \in E \quad (\eta_e) \tag{4.10}$$

$$w_i \leq y_i \qquad \forall i \in V \setminus \{d\} \tag{4.11}$$

$$x \in \mathbb{B}^{|E|}, y, w \in \mathbb{B}^{|V|} \tag{4.12}$$

$$\lambda_p \geq 0, \qquad \forall p \in \mathcal{P}_1 + \mathcal{P}_2. \tag{4.13}$$

As Restrições (4.5) obrigam a escolha de um ciclo de *backbone*, enquanto as Restrições (4.6) e (4.7) fazem com que cada vértice seja coberto pelo *backbone* ou por um ciclo de acesso. As Restrições (4.8) limitam o número de ciclos de acesso ligados ao mesmo *hub* a k e definem o valor das variáveis w para os *hubs*. A Restrição (4.9), a qual chamamos de **Propriedade do ring-ring** (veja explicação a seguir), é usada junto com os dois grupos de restrições anteriores para impedir que dois ciclos tenham uma intersecção maior do que um vértice. As Restrições (4.10) são restrições de aresta para as variáveis x e λ. Por fim, as Restrições (4.11) garantem que todos os *hubs* estejam no *backbone*, as Restrições (4.12) são restrições de integralidade e (4.13) são as restrições de não negatividade.

Propriedade do ring-ring. Seja $R = (V, E)$ uma rede *ring/ring* com n vértices, m arestas e um total de c ciclos. Seja $V_B \subset V$ o conjunto de vértices pertencentes

ao *backbone* e $V_S \subset V$ o conjunto de vértices pertencentes apenas a redes de acesso, com $V_B \cup V_S = V$ e $V_B \cap V_S = \emptyset$.

Denotamos por $\delta(i)$ as arestas de R incidentes no vértice i e por c_i o número de ciclos de R, incluindo o *backbone*, que contém o vértice i.

Uma vez que cada ciclo que contém um vértice i tem exatamente duas arestas incidentes em i, temos que:

$$|\delta(i)| = 2c_i. \tag{4.14}$$

Portanto,

$$c_i = |\delta(i)|/2. \tag{4.15}$$

Podemos observar que cada vértice em V_B participa de pelo menos um ciclo, o *backbone*, mais possivelmente um número de redes de acesso. Sabemos pela definição de um *ring-ring* que cada rede de acesso contém exatamente um vértice do *backbone*. Portanto, podemos obter o número de redes de acesso pela soma dos valores $(c_i - 1)$ (descontamos o *backbone*) para todo vértice i em V_B, como apresentado abaixo:

$$c - 1 = \sum_{i \in V_B} (c_i - 1) \tag{4.16}$$

$$c - 1 = \sum_{i \in V_B} (|\delta(i)|/2 - 1) \tag{4.17}$$

$$c - 1 = (1/2) \sum_{i \in V_B} (|\delta(i)| - 2) \tag{4.18}$$

Já os vértices de V_S participam de apenas um ciclo em uma rede *ring-ring*, que é a sua rede de acesso. Portanto:

$$0 = \sum_{i \in V_S} (c_i - 1) = (1/2) \sum_{i \in V_S} (|\delta(i)| - 2) \tag{4.19}$$

Somando as equações, concluimos que:

$$c - 1 + 0 = (1/2)(\sum_{i \in V_B} (|\delta(i)| - 2) + (1/2) \sum_{i \in V_S} (|\delta(i)| - 2)) \tag{4.20}$$

$$2c - 2 + 0 = \sum_{i \in V_B} (|\delta(i)| - 2) + \sum_{i \in V_S} (|\delta(i)| - 2) \tag{4.21}$$

$$2c - 2 = \sum_{i \in V} (|\delta(i)| - 2) \tag{4.22}$$

$$2c - 2 = \sum_{i \in V} (|\delta(i)|) - 2n \tag{4.23}$$

$$\sum_{i \in V} (|\delta(i)|) - 2c = 2n - 2 \tag{4.24}$$

Substituindo 4.14 em 4.24, obtemos:

$$\sum_{i \in V} (2c_i) - 2c = 2n - 2 \tag{4.25}$$

$$\sum_{i \in V} c_i - c = n - 1. \tag{4.26}$$

Incluindo esta propriedade como restrição no modelo, garantimos que um vértice não esteja em mais de uma rede de acesso. A Figura 4.4 mostra uma rede que poderia ser gerada sem essa restrição. Esta rede possui 7 vértices e 3 ciclos, mas o valor de $\sum_{i \in V} c_i = 10$. Portanto, $10 - 3 = 7 \neq 6$.

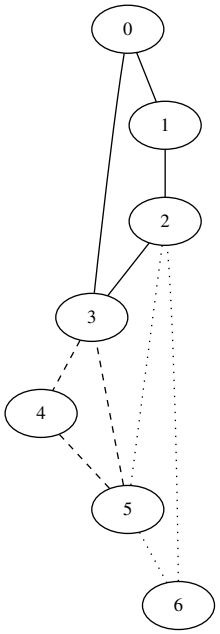

Figura 4.4: Exemplo de rede que viola a restrição.

A Formulação (F1) modela o RRP, conforme a Definição 4.1, que foi adotada neste trabalho. Para modelar o RRP, segundo as Definições 4.2 e 4.3, pequenas alterações na Formulação (F1) são necessárias. Essas alterações são discutidas em seguida.

Inicialmente, considere que $a_i^p = 1$ apenas quando o vértice i pertence ao ciclo p mas não é o *hub*. Além disso, ao invés de considerarmos o conjunto de arestas E, consideramos o conjunto A de arestas orientadas tal que, se $(i, j) \in E$ então $(i, j), (j, i) \in A$, ou seja, A é uma versão orientada de E.

A formulação em PLI para modelar o RRP segundo a Definição 4.2, considerada em Rodríguez-Martín et al. (2016a), é apresentada a seguir.

$$(F2)\ \min \sum_{p\in\mathcal{P}_1} b_p\lambda_p + \sum_{p\in\mathcal{P}_2} c_p\lambda_p + \sum_{i\in V\setminus\{d\}} f_i y_i \tag{4.27}$$

$$s.a. \sum_{p\in\mathcal{P}_1} \lambda_p = 1 \qquad (\pi) \tag{4.28}$$

$$\sum_{p\in\mathcal{P}_1} a_i^p\lambda_p = y_i \qquad \forall i \in V\setminus\{d\} \quad (\alpha_i) \tag{4.29}$$

$$\sum_{p\in\mathcal{P}_2} a_i^p\lambda_p = 1 - y_i \qquad \forall i \in V\setminus\{d\} \quad (\beta_i) \tag{4.30}$$

$$\sum_{p\in\mathcal{P}_2} h_i^p\lambda_p \leq k y_i \qquad \forall i \in V \quad (\theta_i) \tag{4.31}$$

$$\sum_{p\in\mathcal{P}_1+\mathcal{P}_2}\sum_{i\in V} (a_i^p + h_i^p)\lambda_p = n + \sum_{p\in\mathcal{P}_2} \lambda_p \qquad (\gamma) \tag{4.32}$$

$$\sum_{p\in\mathcal{P}_1+\mathcal{P}_2} a_e^p\lambda_p = x_e \qquad \forall e \in A \quad (\eta_e) \tag{4.33}$$

$$x \in \mathbb{B}^{|A|}, y \in \mathbb{B}^{|V|} \tag{4.34}$$

$$\lambda_p \geq 0, \qquad \forall p \in \mathcal{P}_1 + \mathcal{P}_2. \tag{4.35}$$

Na Formulação (F2), as variáveis de decisão w_i foram suprimidas, uma vez que o custo de instalação de facilidade é considerado a todo vértice da rede de *backbone*, independentemente dele ser ou não um hub, ou seja, neste modelo $w_i = y_i$. As demais restrições são similares às da Formulação (F1), por exemplo a substituição das inequações (4.7) pelas restrições de igualdade (4.30), são consequências do fato de desconsideramos os *hubs* do vetor característico a_p de um ciclo p.

Desta vez, para modelar o RRP considerando a Definição (4.3), aquela usada por Rodríguez-Martín et al. (2016b), temos a seguinte formulação estendida:

$$(F3)\ \min \sum_{p\in\mathcal{P}_1} b_p\lambda_p + \sum_{p\in\mathcal{P}_2} c_p\lambda_p + \sum_{i\in V\setminus\{d\}} f_i y_i \tag{4.36}$$

$$s.a. \quad (4.28), (4.29), (4.30), (4.31), \tag{4.37}$$

$$(4.32), (4.33), (4.34), (4.35) \tag{4.38}$$

$$\sum_{p\in\mathcal{P}_2} h_i^p\lambda_p \geq y_i \qquad \forall i \in V \quad (\tau_i) \tag{4.39}$$

As Restrições (4.39) obrigam que haja pelo menos uma rede de acesso associada a cada *hub*.

Como o número de variáveis λ_p é exponencial, usamos o método da geração de colunas para resolver a relaxação linear de (F1), (F2) e (F3). O problema de *pricing* e os diferentes algoritmos para resolvê-los são discutidos nas próximas seções.

4.3 Problema de pricing

Sejam π, α_i, β_i, θ_i, γ e η_e as variáveis duais associadas às Restrições (4.5) (ou (4.28)), (4.6) (ou (4.29)), (4.7) (ou (4.30)), (4.8) (ou (4.31)), (4.9) (ou (4.32)) e (4.10) (ou (4.33)), respectivamente das Formulações (F1), (F2) ou (F3). Adicionalmente, considere as variáveis duais τ_i associadas às Restrições (4.39) da (F3).

O custo reduzido de uma aresta e em um ciclo p é denotado por $\overline{c}_e$, e é dado conforme segue:

$$\overline{c}_e = c_e - \eta_e \tag{4.40}$$

$$\overline{c}_e = \begin{cases} b_e - \eta_e & \text{, se } p \in \mathcal{P}_1 \\ s_e - \eta_e & \text{, se } p \in \mathcal{P}_2. \end{cases} \tag{4.41}$$

O custo reduzido de um vértice i em um ciclo p é denotado por $\overline{c}_i$, e é dado por:

$$\overline{c}_i = \begin{cases} \alpha_i + \gamma & \text{, se } i \in V \text{ e } p \in \mathcal{P}_1 \\ \beta_i + \gamma & \text{, se } i \in V \text{ e } p \in \mathcal{P}_2. \end{cases} \tag{4.42}$$

Considere que $\alpha_d = 0$, $\beta_d = 0$ e $\beta_i = 0$, se i é o hub de p.

Portanto, o **custo reduzido de cada ciclo** p é:

$$\overline{c}_p = \mu + \sum_{e \in E(p)} \overline{c}_e - \sum_{i \in V(p)} (\overline{c}_i + \nu_i) \tag{4.43}$$

no qual, $\nu_i = \theta_i$ na (F1) e (F2) enquanto na (F3), $\nu_i = \theta_i + \tau_i$, em ambos os casos somente se i é o *hub* de p e $p \in \mathcal{P}_2$, caso contrário $\nu_i = 0$. Temos também que $\mu = -\pi$ se $p \in \mathcal{P}_1$, caso contrário $\mu = \gamma$.

O objetivo do *pricing* é encontrar um ciclo de custo reduzido mínimo, ou seja, o problema de *pricing* associado ao PMR consiste em

$$\min_{p \in \mathcal{P}} \overline{c}_p.$$

Propusemos dois modelos de PLI para o problema de *pricing*, os quais são descritos a seguir.

4.3.1 Formulação do Problema de Pricing Múltiplo

Considere as seguintes variáveis de decisão:

$$x_e = \begin{cases} 1, & \text{se a aresta } e \text{ (orientada) está na rede} \\ 0, & c.c. \end{cases}$$

$y_i = \begin{cases} 1, & \text{se o vértice } i \text{ está na rede} \\ 0, & c.c. \end{cases}$

u_i =rótulo do vértice i (MTZ).

A formulação do problema de *pricing*, para um dado vértice h como *hub* do ciclo, é apresentada a seguir:

$$(F4) \quad z_{pric}(h) = \quad \mu - \nu_h + min \sum_{a \in A} \overline{c}_a x_a + \sum_{i \in V} \overline{c}_i y_i$$

$$s.a. \sum_{(i,j) \in \delta^-(i)} x_{ij} = y_i, \qquad \forall i \in V \qquad (4.44)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} = y_i, \qquad \forall i \in V \qquad (4.45)$$

$$(Q+1)x_{ij} + u_i - u_j \leq Q, \qquad \forall (i,j) \in E, j \neq h \qquad (4.46)$$

$$\sum_{i \in V} y_i \geq 3 \qquad (4.47)$$

$$y_h = 1 \qquad (4.48)$$

$$y_d = 0, \qquad \text{se } h \neq d \qquad (4.49)$$

$$-(Q+1)y_i + u_i \leq 0, \qquad \forall i \in V \qquad (4.50)$$

$$x \in \mathbb{B}^{|A|}, y \in \mathbb{B}^{|V|}, u \in \mathbb{N}^{|V|}. \qquad (4.51)$$

As Restrições (4.44) e (4.45) garantem que exatamente um arco em cada direção incida em cada vértice que pertence ao ciclo. As Restrições (4.46) são as restrições MTZ e a Restrição (4.47) garante que p tenha no mínimo 3 vértices. A Restrição (4.48) garante que o *hub* esteja no ciclo e a Restrição (4.49) impede que o depósito esteja no ciclo caso ele não seja o *hub*. As Restrições (4.50) forçam o rótulo dos vértices que não participam do ciclo a ser zero e impedem que o ciclo tenha mais que Q vértices. As Restrições (4.51) são as restrições de integralidade.

4.3.2 Formulação do Problema de Pricing Único

A formulação do problema de *pricing* apresentada na Seção 4.3.1 exige a escolha prévia de um *hub*. A cada iteração do método de geração de colunas, o problema de *pricing* associado a essa formulação é resolvido n vezes, uma para cada possível *hub*.

Uma formulação que não exige a escolha prévia de um *hub* também foi proposta, permitindo que o problema de *pricing* seja resolvido uma única vez para o *backbone* e uma única vez para as redes de acesso, a cada iteração do método de geração de colunas.

O novo modelo possui um novo grupo de variáveis, além das já definidas.

$$R_i = \begin{cases} 1, & \text{se o vértice } i \text{ é escolhido como hub} \\ 0, & c.c. \end{cases}$$

Segue a formulação:

$$
\begin{aligned}
(F5) \quad z_{pric} = \quad & \mu + min \sum_{a \in A} \overline{c}_a x_a + \sum_{i \in V} \overline{c}_i y_i - \sum_{i \in V} \nu_i R_i \\
& s.a. \sum_{(i,j) \in \delta^-(i)} x_{ij} = y_i, && \forall i \in V && (4.52) \\
& \sum_{(i,j) \in \delta^+(i)} x_{ij} = y_i, && \forall i \in V && (4.53) \\
& (Q+1)x_{ij} + u_i - u_j - (Q+1)R_j \leq Q, && \forall (i,j) \in E && (4.54) \\
& \sum_{i \in V} y_i \geq 3 && && (4.55) \\
& \sum_{i \in V} R_i = 1 && && (4.56) \\
& -(Q-1)y_i + u_i + (Q-1)R_i \leq 0, && \forall i \in V && (4.57) \\
& R_d - y_d = 0 && && (4.58) \\
& x \in \mathbb{B}^{|A|}, R, y \in \mathbb{B}^{|V|}, u \in \mathbb{N}^{|V|}. && && (4.59)
\end{aligned}
$$

As Restrições (4.52), (4.53), (4.54), (4.55), (4.57) e (4.59) têm o mesmo propósito que as restrições correspondentes no modelo anterior.

A Restrição (4.56) garante que apenas um *hub* seja escolhido. A Restrição (4.58) faz com que o depósito seja o *hub* caso ele participe do ciclo.

Um comparativo do desempenho dos algoritmos *branch-and-bound* para resolver cada uma das formulações de *pricing* é apresentado no Capítulo 5.

4.4 Geração de Ciclos Usando GRASP

A base da heurística de *pricing* é a geração de ciclos de custo reduzido negativo usando a metaheurística GRASP. Este método também é utilizado pela heurística primal.

A função recebe como parâmetros o grafo G, uma função de custos associada às arestas, uma lista de vértices de G que podem ser incluídos no ciclo, um vértice inicial (ou um valor que indica que ele deve ser escolhido aleatoriamente), um valor inteiro t que corresponde ao tamanho máximo do ciclo a ser formado e um valor real α, $0.0 \leq \alpha \leq 1.0$, usado como critério de aleatoriedade pelo GRASP, conforme descrito na Seção 3.8.

O vértice inicial é o primeiro vértice corrente i e é removido da lista de vértices válidos. Em cada iteração, os vértices válidos são ordenados de maneira crescente pelo custo da aresta que os liga ao vértice i. O menor custo (*cmin*) e o maior custo (*cmax*) da lista são armazenados. Uma segunda lista, LCR (lista

de candidatos restrita), é criada com todos os vértices cujo custo é menor do que $cmin + \alpha \cdot (cmax - cmin)$. O próximo vértice corrente i é escolhido de forma aleatória entre os vértices da LCR, removido da lista de vértices válidos, e seu custo é acrescentado ao custo total do ciclo. Após as duas primeira iterações, o vértice inicial é devolvido à lista de vértices válidos.

O processo é repetido até que sejam escolhidos no máximo $t-1$ vértices. Caso em alguma iteração o vértice inicial seja escolhido, o ciclo está completo. Caso contrário, o t-ésimo vértice é escolhido de forma semelhante, porém considerando-se os custos como a soma do custo da aresta de cada vértice ao vértice corrente i e o custo da aresta de cada vértice ao vértice inicial.

Com o ciclo completo, é feita uma busca local *2-opt*. O ciclo de menor custo encontrado é retornado.

O tempo gasto pela geração de ciclos com o GRASP é $nruns \cdot (Q \cdot |V| + Q^2 + nsols)$ quando o tamanho dos ciclos é limitado a Q. Como $Q \leq |V|$, a complexidade assintótica é $O(|V|^2)$.

4.5 Heurística Primal

Foi implementada uma heurítica primal de arredondamento, descrita no Algoritmo 3. Ela utiliza o GRASP para escolher os ciclos que devem ser adicionados à solução.

A função *limpaCands* (linha 10), chamada após a escolha do *backbone*, remove de cands as variáveis correspondentes a ciclos do tipo *backbone* e a ciclos secundários cujo *hub* não esteja no *backbone*.

A função *criaLCR*, responsável por criar a lista de candidatos restrita, é descrita no Algoritmo 4.

A função *varValida* verifica se uma determinada variável pode ser acrescentada à solução corrente. Para isso, o *hub* de seu ciclo deve ter menos de k ciclos secundários, e ele não pode ter nenhum vértice (além do *hub*) já coberto na solução corrente. Essa função é chamada durante a criação da LCR (linha 17) e garante que apenas variáveis que podem ser acrescentadas à solução corrente sejam consideradas.

Caso a heurística falhe em encontrar uma solução válida entre as variáveis existentes, chamamos *buscaVariaveis* (linha 20), que cria variáveis novas para cobrir os vértices ainda não cobertos, utilizando a geração de ciclos apresentada na Seção 4.4.

Algoritmo 3: Heurística primal

Data: $\lambda, valores$
Result: $solucao$

```
1  solucao ← ∅;
2  backbone = NULL;
3  cands = λ;
4  LCR ← criaLCR(cands, valores, 0, alpha, solucao, backbone);
5  nlcr ← tamanho(LCR);
6  backbone ← LCR[rand()%nlcr];
7  solucao ← solucao ∪ backbone;
8  LCR ← criaLCR(cands, valores, 1, alpha, solucao, backbone);
9  nlcr ← tamanho(LCR);
10 limpaCands(cands, solucao, backbone);
11 while vertices(solucao) ≠ vertices(G)  ou  nlcr > 0 do
12 |   ring ← LCR[rand()%nlcr];
13 |   solucao ← solucao ∪ ring;
14 |   LCR ← criaLCR(cands, valores, 1, alpha, solucao, backbone);
15 |   nlcr ← tamanho(LCR);
16 end
17 if vertices(solucao) = vertices(G) then
18 |   return solucao;
19 end
20 buscaVariaveis(solucao, backbone);
```

4.6 Heurística de Pricing

A heurística de *pricing* consiste em chamar a função de criação de ciclos, primeiro de forma a gerar candidatos a *backbone* e depois de forma a gerar candidados a ciclos secundários. Em cada modo, os respectivos custos reduzidos das arestas são usados. Além disso, no modo *backbone* o depósito é escolhido como vértice inicial, enquanto no modo ciclo secundário o vértice inicial é obtido aleatoriamente e o depósito é excluído da lista de vértices válidos.

Cada modo é executado múltiplas vezes, com os ciclos de menor custo reduzido obtidos ao final sendo incluídos no PMR. O número de execuções e de ciclos salvos é definido previamente. Os valores testados e seus resultados são detalhados no Capítulo 5.

4.7 Problema de Pricing Relaxado

Para acelerar a execução do programa, o problema de *pricing* pode ser substituído por uma relaxação, ou seja, um problema mais simples que forneça um bom limitante para o problema original.

Uma possível relaxação para o problema de *pricing* é utilizando o ng-route,

Algoritmo 4: criaLCR

```
   Data: cands, valores, tipo, alpha, solucao, backbone
   Result: LCR
 1 LCR ← ∅;
 2 if tipo=0 then
 3 |   i ← 0;
 4 |   vmaior = maiorValor(cands, valores, tipo);
 5 |   vmenor = menorValor(cands, valores, tipo);
 6 |   while i < tam(lambda) do
 7 |   |   if valores[cands[i]] >= vmenor + alpha * (vmaior − vmenor)  e
   |   |   tipo(cands[i]) = tipo then
 8 |   |   |   LCR ← LCR ∪ cands[i];
 9 |   |   end
10 |   |   i ← i + 1;
11 |   end
12 end
13 i ← 0;
14 vmaior = maiorValor(cands, valores, tipo);
15 vmenor = menorValor(cands, valores, tipo);
16 while i < tam(lambda) do
17 |   if valores[cands[i]] >= vmenor + alpha * (vmaior − vmenor)  e
   |   varValida(cand[i], tipo, solucao, backbone) then
18 |   |   LCR ← LCR ∪ cands[i];
19 |   end
20 |   i ← i + 1;
21 end
```

descrito na Seção 3.7. A solução do *ng*-route é um ciclo de tamanho máximo Q, podendo haver repetição de vértices. Portanto, nem todos os ciclos gerados serão válidos para o RRP.

Os conjuntos $NG(i)$ de cada vértice i são formados pelos Δ vértices mais próximos a i.

O *ng*-route é resolvido com programação dinâmica, com cada caminho parcial p gerado sendo armazenado na forma de um *label* que contém o último vértice de p ($v(p)$), o número de vértices no caminho até o momento ($\bar{c}(p)$), o conjunto $\Pi(p)$ contendo os vértices para os quais o caminho não pode ser estendido, e o ponteiro para o caminho que foi estendido e gerou p ($ant(p)$).

Durante a geração e expansão dos caminhos, alguns caminhos são dominados, ou seja, são piores que outro caminho já existente, independentemente da extensão escolhida. Um caminho p' é **dominado** por p se todas as condições abaixo (chamadas regras de dominância) são satisfeitas:

(i) $v(p) = v(p')$;

(ii) $\bar{c}(p) \leq \bar{c}(p')$;

(iii) $\Pi(p) \subseteq \Pi(p')$;

(iv) $q(p) \leq q(p')$.

Caminhos dominados são removidos, agilizando o processo.

Para usar o problema de *pricing* relaxado, consideramos, nos modelos $(F1)$, $(F2)$ e $(F3)$, que a_i^p é a quantidade de vezes que o vértice i participa do ciclo p ao invés de ser uma constante binária. Nenhuma alteração adicional é necessária nos modelos. Note que soluções inválidas, em que $a_i^p > 1$, são ignoradas pelas soluções inteiras.

4.8 Heurísticas de Pricing Relaxado

Para melhorar o desempenho do *pricing* relaxado, foram implementadas duas opções de heurística.

A primeira é uma heurística de *bucket pruning*. Apresentada por Fukasawa et al. (2006) para a obtenção de *q-routes* para o problema de roteamento capacitado de veículos, ela consiste na limitação do número de *labels* a serem armazenados na construção de rotas ou ciclos.

Como a heurística limita a quantidade de *labels* armazenados em cada *bucket*, ela pode acabar falhando por não ter armazenado o *label* que produziria um ciclo de custo reduzido negativo. Nestes casos, é necessário que o *pricing* relaxado seja executado novamente, desta vez sem limitar o tamanho do *bucket*, para certificar que nenhum ciclo de custo reduzido negativo realmente exista.

A segunda heurística consiste em tornar a regra de dominância mais agres-

siva. Em particular, a condição $\Pi(p) \subseteq \Pi(p')$ é removida. Isso faz com que mais *labels* sejam dominados, diminuindo o número de candidatos a serem testados, mas também podendo eliminar o ciclo de custo reduzido ótimo. Como na outra heurística, também é necessário executar uma chamada adicional ao *pricing* relaxado, desta vez desabilitando a regra de dominância agressiva.

CAPÍTULO

5

Experimentos Computacionais

Neste capítulo descrevemos os experimentos computacionais que foram conduzidos para avaliar a qualidade dos limitantes duais gerados pelo modelo proposto para o RRP e o desempenho dos algoritmos desenvolvidos neste trabalho. No decorrer deste capítulo, são descritos os experimentos que foram realizados para fazer a calibragem do algoritmo e as tomadas de decisão em relação aos diferentes algoritmos para resolver o problema de *pricing*. Ao final do capítulo, uma análise geral é apresentada em relação à melhor configuração calibrada e uma comparação é feita com os resultados reportados na literatura.

5.1 Ambiente Computacional

Os experimentos foram realizados em um computador *desktop* i7-4790 (3.6GHz) com 32 GB de RAM e com o sistema operacional Linux e *multi-thread* desabilitado. Todos os algoritmos foram implementados na linguagem C usando compilador gcc 4.9.2 e a biblioteca de otimização SCIP (Gamrath et al. (2016)) com o IBM Cplex (v12.6.1.0) (Cplex (2015)) sendo usado como resolvedor de programação linear.

Em todos os testes realizados, usamos as configurações padrão do SCIP, mas desabilitando as heurísticas e os cortes genéricos fornecidos pela biblioteca, com o intuito de avaliar isoladamente o impacto das diferentes implementações que foram propostas neste trabalho.

5.2 Instâncias

Para analisar o modelo, usamos dois grupos de instâncias: Grupo A e Grupo B. O primeiro grupo contém 39 instâncias derivadas das instâncias do CRTP Hill and Voß (2016) enquanto o segundo grupo contém 144 instâncias já utilizadas em trabalhos relacionados ao RRP na literatura (Rodríguez-Martín et al. (2016a,b)). O Grupo B é dividido em duas classes, denominadas I e II.

A calibragem dos parâmetros e a análise das heurísticas foram realizadas usando-se a Definição 4.1 para o RRP.

Para a escolha dos parâmetros, utilizamos um subconjunto de instâncias contendo 9 instâncias do Grupo A e 18 instâncias do Grupo B, totalizando 27 instâncias, que chamaremos de **instâncias teste**. A Tabela 5.1 apresenta os dados dessas instâncias escolhidas. A coluna $|V|$ mostra o número de vértices do grafo de entrada e a coluna $|E|$, o número de arestas. A coluna Q indica o tamanho máximo de cada *ring* e a coluna k o número máximo de *rings* por *hub*. As colunas classe e f_i referem-se à classe I ou II e ao intervalo para o valor do custo de instalação de facilidade, as quais estão definidas apenas para as instâncias do Grupo B. Essas informações permitem relacionar os nomes abreviados das instâncias àqueles definidos na literatura.

5.3 Análise dos Modelos de Pricing

Primeiramente, realizamos experimentos para determinar o impacto na escolha da formulação de *pricing*. Na Tabela 5.2 apresentamos os resultados obtidos com o algoritmo *branch-and-price*, resolvendo-se o problema de *pricing* usando o modelo $(F4)$, descrito na Seção 4.3.1 e denominado modelo de *pricing* múltiplo, e com o modelo $(F5)$, descrito na Seção 4.3.2, o qual chamamos de *pricing* único. Os experimentos foram realizados nos dois grupos de instâncias teste. Nestes experimentos, somente a heurística de *pricing* foi ativada e com os seguintes parâmetros iniciais, os quais foram definidos de forma empírica: $\alpha = 0{,}5$, $nruns = 100$ e $nsols = 10$.

Nesta análise, comparamos o desempenho do algoritmo avaliando o *gap* de dualidade e o tempo total gasto pelo algoritmo. O comparativo é apresentado na Tabela 5.2. A coluna `gap` apresenta o *gap* de dualidade que é dado por $(ub - lb)/lb$, sendo ub o valor da solução ótima ou da melhor solução encontrada pelo algoritmo e lb o melhor limitante dual. A coluna `Tempo` denota o tempo, em segundos, gasto pelo algoritmo. Um limite de tempo para a execução do algoritmo em cada instância foi de 15 minutos. O símbolo $-$ é usado quando o tempo limite é atingido ou quando o algoritmo terminou sem encontrar um limitante dual no nó raiz. O símbolo $*$ indica que nenhum *gap* de dualidade

instância	classe	f_i	$\|V\|$	$\|E\|$	Q	k
A22	-	-	19	171	7	3
A23	-	-	19	171	5	4
A24	-	-	19	171	4	5
A25	-	-	38	703	14	3
A26	-	-	38	703	11	4
A27	-	-	38	703	9	5
A28	-	-	57	1596	21	3
A29	-	-	57	1596	16	4
A30	-	-	57	1596	13	5
C04-II.2	II	[1,500]	15	105	12	2
C05-II.2	II	[1,500]	15	105	4	2
C06-II.2	II	[1,500]	15	105	8	2
D10-II.2	II	[500,1000]	20	190	10	2
D11-II.2	II	[500,1000]	20	190	15	2
D12-II.2	II	[500,1000]	20	190	5	2
E16-II.2	II	[1200,1700]	30	435	15	2
E17-II.2	II	[1200,1700]	30	435	23	2
E18-II.2	II	[1200,1700]	30	435	8	2
C04-I.2	I	[1,500]	15	105	12	2
C05-I.2	I	[1,500]	15	105	4	2
C06-I.2	I	[1,500]	15	105	8	2
D10-I.2	I	[500,1000]	20	190	10	2
D11-I.2	I	[500,1000]	20	190	15	2
D12-I.2	I	[500,1000]	20	190	5	2
E16-I.2	I	[1200,1700]	30	435	15	2
E17-I.2	I	[1200,1700]	30	435	23	2
E18-I.2	I	[1200,1700]	30	435	8	2

Tabela 5.1: Dados das instâncias escolhidas para os testes de calibragem de parâmetros.

está disponível devido ao limite de tempo.

instância	$\|V\|$	Q	k	*pricing* múltiplo gap	*pricing* múltiplo Tempo	*pricing* único gap	*pricing* único Tempo
A22	19	7	3	**49,95**	-	**49,95**	-
A23	19	5	4	23,95	-	**19,13**	-
A24	19	4	5	19,88	-	**17,42**	-
A25	38	14	3	*	-	*	-
A26	38	11	4	*	-	*	-
A27	38	9	5	*	-	*	-
A28	57	21	3	*	-	*	-
A29	57	16	4	*	-	*	-
A30	57	13	5	*	-	*	-
C04-II.2	15	12	2	**164,20**	-	*	-
C05-II.2	15	4	2	**0,00**	668,62	**0,00**	**167,82**
C06-II.2	15	8	2	68,63	-	**63,85**	-
D10-II.2	20	10	2	*	-	*	-
D11-II.2	20	15	2	*	-	*	-
D12-II.2	20	5	2	3,92	-	**0,00**	**768,18**
E16-II.2	30	15	2	*	-	*	-
E17-II.2	30	23	2	*	-	*	-
E18-II.2	30	8	2	*	-	*	-
C04-I.2	15	12	2	4,92	-	**0,00**	**750,73**
C05-I.2	15	4	2	**0,00**	92,44	**0,00**	**31,55**
C06-I.2	15	8	2	**23,74**	-	23,74	-
D10-I.2	20	10	2	160,3	-	**16,29**	-
D11-I.2	20	15	2	*	-	**155,02**	-
D12-I.2	20	5	2	7,81	-	**0,00**	**823,06**
E16-I.2	30	15	2	*	-	*	-
E17-I.2	30	23	2	*	-	*	-
E18-I.2	30	8	2	*	-	**117,10**	-

Tabela 5.2: Resultados na execução dos modelos *pricing* múltiplo e *pricing* único.

Observando a tabela, percebemos que o modelo *pricing* único obteve *gap* para mais instâncias, *gaps* menores para a maioria das instâncias em que ambos os modelos conseguiram *gap*, e tempo de execução menor quando os *gaps* são os mesmos. Além disso, 5 instâncias foram resolvidas na otimalidade, enquanto o *pricing* múltiplo só resolveu 2. Portanto, concluímos que o modelo *pricing* único obteve melhor desempenho, sendo assim o escolhido para as próximas etapas.

5.4 Análise dos Parâmetros da Geração de Ciclos com GRASP

O parâmetro α da heurística de geração de ciclos usando GRASP, que é usada pela heurística de *pricing* e pela heurística primal, foi avaliado para 5 valores variando entre 0,0 e 1,0. Os resultados deste experimento são apresentados na Tabela 5.3. A execução em cada instância foi limitada a 30 minutos e apenas ao nó raiz. A coluna resolvidas apresenta o número de instâncias para as quais um valor de gap foi obtido pela execução.

α	resolvidas	melhor tempo	média tempo
0,0	**15**	2	314,56
0,1	14	1	261,08
0,3	**15**	4	269,21
0,5	14	**6**	**242,65**
1,0	14	2	300,89

Tabela 5.3: Resumo dos testes para definição de α.

Como podemos observar na Tabela 5.3, os testes para os valores 0,0 e 0,3 obtiveram os melhores resultados quando se trata de número de instâncias resolvidas, e entre eles 0,3 conseguiu o melhor tempo em mais instâncias. Portanto, definimos $\alpha = 0{,}3$.

Em seguida, avaliamos a escolha do parâmetro *nruns*, que define o número de iterações da heurística de geração de ciclos usando GRASP. Para este experimento, avaliamos os seguintes valores para *nruns*: 50, 100, 200, 500 e 1.000. Os resultados deste experimentos estão resumidos na Tabela 5.4. Novamente, a execução em cada instância foi limitada a 30 minutos e apenas ao nó raiz.

nruns	resolvidas	melhor tempo	média tempo
50	14	2	169,51
100	14	**5**	165,07
200	13	2	247,24
500	**15**	1	171,83
1.000	14	4	**135,96**

Tabela 5.4: Resumo dos testes para definição de *nruns*.

O valor escolhido para *nruns* foi 500, por resolver mais instâncias.

O parâmetro *nsols*, que indica o número de melhores soluções que devem ser armazenadas e devolvidas pela heurística de geração de colunas usando GRASP, foi avaliado para valores relativos a *nruns*: 10%, 20%, 30%, 40%, 50% e 100% do valor de *nruns*. A execução em cada instância foi limitada a 30 minutos e apenas ao nó raiz.

nsols	resolvidas	melhor tempo	média tempo
50	**15**	1	304,35
100	14	2	247,32
150	14	2	**236,58**
200	14	3	246,79
250	14	**5**	250,71
500	14	1	331,89

Tabela 5.5: Resumo dos testes para definição de *nsols*.

A Tabela 5.5 sumariza os resultados obtidos neste experimento.

O valor escolhido foi 50, também por resolver o maior número de instâncias.

5.5 Análise da Heurística de Pricing

Definidos os parâmetros do algoritmo de geração de ciclos usando GRASP, realizamos experimentos para avaliar o impacto no desempenho do algoritmo *branch-and-price* ao habilitar a heurística de *pricing*.

A Tabela 5.6 compara os resultados do algoritmo usando o modelo *pricing* único sem a heurística e com a heurística de *pricing* com os parâmetros definidos após a calibragem. Nessa tabela, a coluna `tempo` representa o tempo total gasto pelo algoritmo em segundos e a coluna `speedup` apresenta a razão entre os tempos da versão com a heurística e sem a heurística de *pricing*. Um -"indica as instâncias para as quais o *speedup* não é definido. A execução em cada instância foi limitada a 30 minutos e apenas ao nó raiz.

Percebemos uma melhoria significativa com o uso da heurística de *pricing* em relação aos resultados sem heurística, o melhor obtido com o uso de $\alpha = 0,3$, $nruns = 500$ e $nsols = 50$.

O modelo com a heurística de *pricing* conseguiu resolver uma instância a mais e melhorou o tempo do nó raiz em 10 das 14 instâncias.

A seguir, a Tabela 5.7 apresenta a melhoria obtida na heurística de *pricing* com os parâmetros calibrados, em relação aos parâmetros iniciais. O tempo limite é de 15 minutos.

5.6 Análise do Problema de Pricing Relaxado

Comparamos então os resultados do modelo com *pricing* único com os do modelo com o *pricing* relaxado no nó raiz usando o *ng-route* na Tabela 5.8. A execução foi feita apenas no nó raiz e a coluna $gap0$ representa o gap no nó raiz, definido por $(ub(best) - lb0)/ub(best)$, onde $ub(best)$ representa o limitante superior da melhor execução. A execução em cada instância foi limitada a 30 minutos e apenas ao nó raiz.

instância	sem heur. tempo	heur. tempo	heur. speedup
A22	567,31	**104,54**	5,43
A23	130,53	**15,31**	8,52
A24	57,28	**5,82**	9,84
A25	-	-	-
A26	-	-	-
A27	-	-	-
A28	-	-	-
A29	-	-	-
A30	-	-	-
C04-II.2	**1355,77**	1491,85	0,91
C05-II.2	19,83	**6,02**	3,29
C06-II.2	730,19	**394,30**	1,85
D10-II.2	-	-	-
D11-II.2	-	**1186,57**	-
D12-II.2	150,47	**107,94**	1,39
E16-II.2	-	-	-
E17-II.2	-	-	-
E18-II.2	-	-	-
C04-I.2	**54,34**	95,10	0,57
C05-I.2	16,72	**7,32**	2,28
C06-I.2	39,78	**26,47**	1,50
D10-I.2	323,89	**269,50**	1,20
D11-I.2	**525,82**	966,33	0,54
D12-I.2	65,90	**12,35**	5,34
E16-I.2	-	-	-
E17-I.2	-	-	-
E18-I.2	**584,26**	758,11	0,77

Tabela 5.6: Resultados na execução do modelo com e sem heurística de *pricing*, apenas no nó raiz.

instância	iniciais *gap*	iniciais Tempo	calibrados *gap*	calibrados Tempo
A22	49,95	-	**49,72**	-
A23	19,13	-	**18,26**	-
A24	17,42	-	**8,97**	-
A25	*	-	*	-
A26	*	-	*	-
A27	*	-	*	-
A28	*	-	*	-
A29	*	-	*	-
A30	*	-	*	-
C04-II.2	*	-	*	-
C05-II.2	0,00	**167,82**	0,00	266,41
C06-II.2	**63,85**	-	68,63	-
D10-II.2	*	-	*	-
D11-II.2	*	-	*	-
D12-II.2	0,00	**768,18**	0,00	832,94
E16-II.2	*	-	*	-
E17-II.2	*	-	*	-
E18-II.2	*	-	*	-
C04-I.2	0,00	750,73	0,00	**585,03**
C05-I.2	0,00	**31,55**	0,00	40,81
C06-I.2	**23,74**	-	**23,74**	-
D10-I.2	**16,29**	-	19,43	-
D11-I.2	**155,02**	-	160,27	-
D12-I.2	0,00	823,06	0,00	**723,13**
E16-I.2	*	-	*	-
E17-I.2	*	-	*	-
E18-I.2	**117,1**	-	**117,1**	-

Tabela 5.7: Resultados da execução da heurística com os parâmetros calibrados e com os parâmetros padrões.

instância	exato		relaxado	
	gap0	Tempo	*gap0*	Tempo
A22-n076-m03	**6,41**	104,54	8,50	**0,36**
A23-n076-m04	**13,94**	15,31	15,17	**0,19**
A24-n076-m05	**15,26**	5,82	18,11	**0,10**
A25-n076-m03	*	-	22,58	88,11
A26-n076-m04	*	-	42,53	28,92
A27-n076-m05	*	-	56,01	13,76
A28-n076-m03	*	-	*	-
A29-n076-m04	*	-	80,49	679,76
A30-n076-m05	*	-	73,32	292,09
C04-II.2	**19,81**	1491,85	22,45	**1,17**
C05-II.2	**12,50**	6,02	15,76	**0,07**
C06-II.2	**15,55**	394,3	18,90	**0,45**
D10-II.2	*	-	5,63	1,90
D11-II.2	**26,26**	1186,57	26,90	**2,81**
D12-II.2	**5,56**	107,94	8,27	**0,22**
E16-II.2	*	-	12,58	23,30
E17-II.2	*	-	48,21	125,15
E18-II.2	*	-	20,73	2,02
C04-I.2	**17,79**	95,10	23,82	**1,32**
C05-I.2	**7,77**	7,32	14,55	**0,08**
C06-I.2	**15,26**	26,47	20,15	**0,4**
D10-I.2	**24,3**	269,5	24,82	**1,83**
D11-I.2	**55,64**	966,33	58,04	**12,75**
D12-I.2	**11,07**	12,35	11,37	**0,17**
E16-I.2	*	-	51,11	25,36
E17-I.2	*	-	187,17	88,82
E18-I.2	**26,69**	758,11	27,25	**2,85**

Tabela 5.8: Resultados da comparação entre o *pricing* exato e o *pricing* relaxado.

Observamos que, nas instâncias resolvidas pelos dois modelos, o *pricing* exato sempre obteve *gap* menor, mas o *pricing* relaxado terminou o nó raiz em tempo expressivamente melhor. Entre as instâncias que o *pricing* exato não conseguiu resolver, apenas uma não foi resolvida pelo *pricing* relaxado. Observa-se assim a vantagem de se usar o *pricing* relaxado.

Em seguida, testamos a variação do parâmetro Δ, utilizado pelo *pricing* relaxado. O tempo limite foi de 15 minutos. Os resultados estão na Tabela 5.9.

	Δ=3			Δ=5			Δ=10		
instância	*gap*	lb	Tempo	*gap*	lb	Tempo	*gap*	lb	Tempo
A22	**0,0**	**163,0**	**8,1**	**0,0**	**163,0**	12,9	**0,0**	**163,0**	34,6
A23	**0,0**	**189,0**	30,0	**0,0**	**189,0**	**24,7**	**0,0**	**189,0**	35,1
A24	**0,0**	**207,0**	**5,9**	**0,0**	**207,0**	8,5	**0,0**	**207**	7,6
A25	**33,1**	196,8	-	33,5	**197,0**	-	35,9	195,7	-
A26	45,5	201,4	-	**29,6**	**203,8**	-	37,3	202,4	-
A27	45,1	212,9	-	**33,5**	**208,2**	-	43,3	210,1	-
A28	*	*	-	*	*	-	*	*	-
A29	69,1	222,9	-	*	*	-	*	*	-
A30	**64,1**	226,7	-	92,2	**228,4**	-	*	*	-
C04-II.2	**0,0**	**785,0**	**178,8**	**0,0**	**785,0**	533,5	**0,0**	**785,0**	386,7
C05-II.2	**0,0**	**1255,0**	2,2	**0,0**	**1255,0**	2,5	**0,0**	**1255**	**0,7**
C06-II.2	**0,0**	**860,0**	**39,4**	**0,0**	**860,0**	109,6	**0,0**	**860,0**	174,9
D10-II.2	**0,0**	**1211,0**	**83,1**	**0,0**	**1211,0**	120,1	**0,0**	**1211,0**	169,9
D11-II.2	**0,0**	**1133,0**	301,6	**0,0**	**1133,0**	**236,4**	5,9	1069,9	*
D12-II.2	**0,0**	**2160,0**	7,5	**0,0**	**2160,0**	**2,3**	**0,0**	**2160,0**	4,1
E16-II.2	**6,7**	**1842,7**	-	10,0	1789,0	-	117,2	1784,5	-
E17-II.2	135,6	**1386,9**	-	**60,9**	1408,1	-	*	*	-
E18-II.2	17,5	3006,8	*	**10,1**	**3186,0**	-	11,5	3126,6	-
C04-I.2	**0,0**	**601,0**	21,2	**0,0**	**601,0**	**18,9**	**0,0**	**601,0**	60,1
C05-I.2	**0,0**	**812,0**	2,2	**0,0**	**812,0**	**0,2**	**0,0**	**812,0**	**0,2**
C06-I.2	**0,0**	**605,0**	7,8	**0,0**	**605,0**	15,1	**0,0**	**605,0**	**7,3**
D10-I.2	**0,0**	**1031,0**	**44,4**	**0,0**	**1031,0**	57,0	**0,0**	**1031,0**	155,2
D11-I.2	**6,2**	**1006,3**	-	22,8	834,6	-	39,41	761,1	-
D12-I.2	**0,0**	**1838,0**	**3,6**	**0,0**	**1838,0**	5,1	**0,0**	**1838,0**	4,9
E16-I.2	54,1	1097,6	-	**47,9**	**1099,6**	-	136,7	1084,9	-
E17-I.2	272,8	758,2	-	272,5	759,0	-	**254,2**	**761,7**	-
E18-I.2	**1,7**	2860,9	-	**1,7**	**2861,6**	-	22,7	2390,4	-

Tabela 5.9: Comparação de desempenho em relação ao valor de Δ.

Neste caso, não se percebe uma melhoria significativa para valores mais altos de Δ. A execução com $\Delta = 3$ encontrou gap para 26 instâncias, enquanto a com $\Delta = 5$ encontrou para 25 e a com $\Delta = 10$ encontrou para 23 instâncias. Foram resolvidas à otimalidade 14 instâncias tanto com $\Delta = 3$ quanto com $\Delta = 5$, enquanto com $\Delta = 10$ foram resolvidas 13. A execução com $\Delta = 3$ conseguiu o melhor tempo entre as três em 7 instâncias. Portanto definimos fixar $\Delta = 3$.

5.7 Análise da Heurística Primal

Escolhido o tipo de *pricing*, agora apresentamos os resultados obtidos ao adicionar a heurística primal na Tabela 5.10. O tempo limite foi de 15 minutos.

instância	sem heur. *gap*	sem heur. Tempo	com heur. *gap*	com heur. Tempo
A22	**0,00**	26,19	**0,00**	**8,06**
A23	**0,00**	30,17	**0,00**	**29,96**
A24	**0,00**	6,12	**0,00**	**5,93**
A25	97,86	*	**33,10**	*
A26	109,88	*	**45,46**	*
A27	104,24	*	**45,13**	*
A28	-	*	-	*
A29	113,27	*	**69,15**	*
A30	106,14	*	**64,07**	*
C04-II.2	**0,00**	845,04	**0,00**	**178,85**
C05-II.2	**0,00**	2,95	**0,00**	**2,22**
C06-II.2	**0,00**	90,42	**0,00**	**39,39**
D10-II.2	**0,00**	94,93	**0,00**	**83,1**
D11-II.2	**0,00**	347,33	**0,00**	**301,60**
D12-II.2	**0,00**	8,11	**0,00**	**7,50**
E16-II.2	10,37	*	**6,75**	*
E17-II.2	**134,69**	*	135,56	*
E18-II.2	**6,89**	*	17,50	*
C04-I.2	**0,00**	24,82	**0,00**	**21,17**
C05-I.2	**0,00**	**2,07**	**0,00**	2,17
C06-I.2	**0,00**	8,17	**0,00**	**7,84**
D10-I.2	**0,00**	**43,33**	**0,00**	44,42
D11-I.2	**3,10**	*	6,23	*
D12-I.2	**0,00**	**2,71**	**0,00**	3,63
E16-I.2	274,52	*	**54,06**	*
E17-I.2	**272,84**	*	**272,84**	*
E18-I.2	**1,19**	*	1,72	*

Tabela 5.10: Resultados da execução sem e com a heurística primal.

Das 14 instâncias resolvidas à otimalidade pelas duas versões, o tempo é menor em 11 na versão com a heurística primal. Das outras instâncias, a versão heurística com primal obteve melhor *gap* em 7, e empatou em 1.

5.8 Heurísticas de Pricing Relaxado

Testamos individualmente as duas heurísticas do *pricing* relaxado. O tempo limite foi de 15 minutos. Os resultados estão na Tabela 5.11.

	sem heur.			heur. 1			heur. 2		
instância	*gap*	lb	Tempo	*gap*	lb	Tempo	*gap*	lb	Tempo
A22	**0,0**	**163,0**	**8,1**	**0,0**	**163,0**	35,0	**0,0**	**163,0**	16,9
A23	**0,0**	**189,0**	**30,0**	**0,0**	**189,0**	36,2	**0,0**	**189,0**	48,7
A24	**0,0**	**207,0**	**5,9**	**0,0**	**207,0**	7,1	**0,0**	**207,0**	6,2
A25	**33,1**	**196,8**	*	100,9	194,1	*	39,4	196,5	*
A26	45,5	**201,4**	*	111,1	200,8	*	**42,4**	200,1	*
A27	**45,1**	**212,9**	*	104,2	207,6	*	55,4	209,9	*
A28	-	-	*	-	-	*	-	-	*
A29	69,2	222,9	*	-	-	*	-	-	*
A30	64,1	**226,7**	*	106,5	226,6	*	**63,4**	226,4	*
C04-II.2	**0,0**	**785,0**	**178,8**	1,6	778,6	*	23,0	777,5	*
C05-II.2	**0,0**	**1255,0**	**2,2**	**0,0**	**1255,0**	3,3	**0,0**	**1255,0**	3,3
C06-II.2	**0,0**	**860,0**	**39,4**	**0,0**	**860,0**	109,5	**0,0**	**860,0**	152,9
D10-II.2	**0,0**	**1211,0**	**83,1**	**0,0**	**1211,0**	132,2	**0,0**	**1211,0**	165,7
D11-II.2	**0,0**	**1133,0**	**301,6**	**0,0**	**1133,0**	477,5	**0,0**	**1133,0**	419,2
D12-II.2	**0,0**	**2160,0**	7,5	**0,0**	**2160,0**	10,2	**0,0**	**2160,0**	**5,5**
E16-II.2	**6,8**	**1842,7**	*	11,6	1772,6	*	10,2	1791,2	*
E17-II.2	135,6	**1386,9**	*	135,6	**1386,9**	*	**50,9**	1382,2	*
E18-II.2	17,5	3006,8	*	**9,8**	**3185,3**	*	19,2	3026,0	*
C04-I.2	**0,0**	**601,0**	**21,2**	**0,0**	**601,0**	33,8	**0,0**	**601,0**	33,6
C05-I.2	**0,0**	**812,0**	2,2	**0,0**	**812,0**	2,4	**0,0**	**812,0**	**0,7**
C06-I.2	**0,0**	**605,0**	**7,8**	**0,0**	**605,0**	10,5	**0,0**	**605,0**	9,5
D10-I.2	**0,0**	**1031,0**	**44,4**	**0,0**	**1031,0**	61,5	**0,0**	**1031,0**	69,9
D11-I.2	**6,2**	**1006,3**	*	26,7	842,7	*	25,2	846,6	*
D12-I.2	**0,0**	**1838,0**	3,6	**0,0**	**1838,0**	**3,5**	**0,0**	**1838,0**	4,5
E16-I.2	**54,1**	1097,6	*	281,4	**1109,3**	*	59,2	**1109,3**	*
E17-I.2	**272,8**	**758,2**	*	**272,8**	**758,2**	*	275,7	752,5	*
E18-I.2	**1,7**	**2860,9**	*	2,3	2860,0	*	14,5	2574,7	*

Tabela 5.11: Resultados da execução das instâncias de teste sem heurísticas de *pricing* relaxado, com a heurística 1 (*bucket pruning*, e com a heurística 2 (regra de dominância mais agressiva).

Diferentemente das outras heurísticas, ambas as heurísticas de *pricing* relaxado não apresentaram melhorias significativas, mostrando que o custo de suas operações não resultou em melhoria nos limitantes obtidos. Um maior número de instâncias foi resolvido à otimalidade, assim como mais instâncias foram resolvidas em tempo menor.

5.9 Análise Geral

A Tabelas 5.12, 5.13 e 5.14 apresentam os resultados resumidos que foram obtidos na execução do modelo com os parâmetros calibrados em todas as instâncias, divididas por grupo e classe. Em cada tabela, as instâncias foram agrupadas por número de vértices, e valores de k. Os valores apresentados na coluna Q são a média dos valores de Q nas instâncias consideradas. A coluna "total" apresenta o número total de instâncias que se enquadram nesses valores, seguido pelo número de instâncias resolvidas à otimalidade. O *gap* médio é calculado considerando apenas as instâncias não resolvidas à otimalidade. Já o tempo médio considera apenas as instâncias resolvidas à otimalidade, assim como o $gap0$, definido por $(ub - lb0)/ub$. O tempo limite foi de 15 minutos.

Nosso algoritmo consegue resolver a maior parte das instâncias com menos de 30 vértices à otimalidade, com *gap0* menor para valores menores de k. O tempo gasto, no entanto, parece estar relacionado ao valor de Q, sendo menor para valores menores no Grupo A. O valor de *gap* obtido aumenta conforme o tamanho das instâncias aumenta.

$\|V\|$	k	Q	total	*gap*	*gap0*	tempo médio
$\|V\| \leq 26$	3	7,8	5(3)	17,2	10,3	27,4
	4	5,6	5(3)	18,2	14,9	19,5
	5	4,6	5(3)	11,3	15,8	4,7
$26 < \|V\| \leq 51$	3	16,5	4(0)	37,0	*	-
	4	12,5	4(0)	47,6	*	-
	5	10,5	4(0)	50,3	*	-
$51 < \|V\| \leq 101$	3	28,8	4(0)	*	*	-
	4	21,5	4(0)	80,5	*	-
	5	17,5	4(0)	73,0	*	-
All			9/39	39,3	13,7	17,2

Tabela 5.12: Resumo do desempenho nas instâncias do Grupo A.

5.10 Análise com Resultados da Literatura

Para avaliar a qualidade das soluções geradas e o desempenho do algoritmo *branch-and-price* proposto com os resultados reportados na literatura, usamos as formulações $(F2)$ e $(F3)$, descritas na Seção 3 para modelar o RRP segundo as Definições 4.2 e 4.3.

Neste experimentos, usamos a configuração final dos parâmetros e dos algoritmos descritos nas seções anteriores. Os resultados do algoritmo *branch-and-price* foram comparados com aqueles reportados por Rodríguez-Martín

$\|V\|$	k	Q	total	*gap*	*gap0*	tempo médio
15	1	10	6(6)	-	15,1	18,5
20	1	10	9(9)	-	16,3	166,3
30	1	15,3	9(0)	76,0	*	-
40	1	20	9(0)	115,1	*	-
15	2	8	9(9)	-	23,5	15,6
20	2	10	9(8)	5,8	18,3	85,0
30	2	15,3	9(0)	59,7	*	-
40	2	20	9(0)	72,3	*	-
All			32/69	78,2	18,6	75,9

Tabela 5.13: Resumo do desempenho nas instâncias da Classe I do Grupo *B*.

$\|V\|$	k	Q	total	*gap*	*gap0*	tempo médio
15	1	10	6(6)	-	10,5	76,7
20	1	10	9(7)	2,0	13,2	129,4
30	1	15,3	9(0)	43,4	*	-
40	1	20	9(0)	59,3	*	-
15	2	8	9(9)	-	19,6	162,5
20	2	10	9(9)	-	11,7	111,3
30	2	15,3	9(1)	25,9	12,4	681,2
40	2	20	9(0)	63,1	*	-
All			32/69	46,0	14,0	141,0

Tabela 5.14: Resumo do desempenho nas instâncias da Classe II do Grupo *B*.

et al. (2016a) e Rodríguez-Martín et al. (2016b) para as Definições 4.2 e 4.3, respectivamente. Avaliamos a versão básica 1 (denotada por `Basic 1`) dos algoritmos *branch-and-cut* apresentados nestes trabalhos, uma vez que essa versão não inclui desigualdades válidas. A separação de cortes é usada apenas para incluir as desigualdades do modelo que são necessárias para garantir a viabilidade da solução inteira. Apresentamos também os resultados da versão completa destes algoritmos (denotada por `Complete`).

Os resultados são apresentados nas Tabelas 5.15 e 5.16. O tempo apresentado foi corrigido para compensar a diferença entre as máquinas onde os testes foram realizados. Um fator de $3{,}6/3{,}8$ foi aplicado nos tempos gastos pelo nosso algoritmo. Nestas tabelas, diferentemente das anteriores, o *gap0* é calculado como $(opt - lb0)/opt$, onde opt é a solução ótima, da mesma forma que é calculado nos trabalhos citados, para que seja possível a comparação.

				Basic 1		*Complete*	
instância	Q	*gap0*	tempo	*gap0*	tempo	*gap0*	tempo
C01-I.1	12	1,73	18,08	1,57	3,39	0,85	0,22
C02-I.1	4	9,82	5,82	14,89	200,96	6,78	2,57
C03-I.1	8	1,45	2,09	1,51	3,60	0,97	0,16
C01-II.1	12	0,04	0,88	0,22	2,40	0,00	0,14
C02-II.1	4	3,71	1,52	8,13	178,08	2,51	1,97
C03-II.1	8	2,27	-	4,22	12,68	2,88	0,73

Tabela 5.15: Comparação de desempenho do algoritmo *branch-and-price*, usando a Definição 4.2, com o algoritmo *branch-and-cut* apresentado em Rodríguez-Martín et al. (2016a), em suas versões básica e completa.

Na Definição 4.2, comparada com o *Basic 1*, nossa formulação obteve *gap0* menor em 5 das 6 instâncias analisadas, com tempo melhor para 4 delas. A diferença de tempo é especialmente expressiva nas instâncias com valor menor de Q, nas quais o algoritmo *branch-and-but Basic 1* gastou mais tempo. Ao fazermos a comparação com o *branch-and-cut Complete*, nossa formulação obteve *gap0* menor em apenas uma instância, e gastou mais tempo em todas.

Nossa formulação também obteve *gap* menor e tempo melhor na maior parte das instâncias analisadas para o *Basic 1* da Definição 4.3, também obtendo melhoria significativa nos tempos das instâncias da Classe I com poucos vértices e valor baixo de Q. Além disso, o algoritmo *branch-and-price* foi capaz de resolver todas as instâncias no tempo limite enquanto o algoritmo *branch-and-cut Basic 1* não resolveu três das instâncias com 20 vértices e $Q = 5$. O *branch-and-cut Complete* obteve melhor tempo para a maior parte das instâncias do que o *branch-and-price*, mas o *branch-and-price* obteve *gap0* melhor em 18 das instâncias, enquanto o *branch-and-cut Complete* obteve *gap0* melhor em 16 instâncias (nas outras 2 instâncias testadas houve empate).

				Basic 1		*Complete*	
instância	*Q*	*gap0*	tempo	*gap0*	tempo	*gap0*	tempo
C04-I.2	12	1,03	1,50	0,98	4,48	0,00	0,25
C05-I.2	4	2,72	0,69	5,01	150,12	3,76	1,84
C06-I.2	8	1,35	2,56	1,44	6,08	0,00	0,19
C10-I.2	10	0,29	2,13	0,32	15,87	0,00	0,70
C11-I.2	15	0,44	15,3	1,39	22,74	1,18	0,34
C12-I.2	5	2,29	10,48	5,01	1042,98	4,53	42,59
D04-I.2	12	0,02	0,35	0,00	2,87	0,00	0,08
D05-I.2	4	2,07	3,23	3,12	249,24	2,61	4,59
D06-I.2	8	0,02	0,15	0,00	3,67	0,00	0,06
D10-I.2	10	0,18	4,03	0,30	12,98	0,14	0,72
D11-I.2	15	0,23	15,07	0,50	14,54	0,46	0,27
D12-I.2	5	1,46	110,26	2,82	341,35	2,22	11,89
E04-I.2	12	0,21	2,15	0,19	5,57	0,00	0,19
E05-I.2	4	0,57	0,91	1,11	44,01	0,76	1,83
E06-I.2	8	0,27	1,58	0,10	8,83	0,00	1,50
E10-I.2	10	0,07	2,38	0,29	18,60	0,28	0,30
E11-I.2	15	0,11	21,25	0,32	25,19	0,30	0,25
E12-I.2	5	0,61	12,79	1,01	1248,91	0,99	9,72
C04-II.2	12	0,63	1,23	0,00	2,89	0,00	0,17
C05-II.2	4	0,00	0,06	2,59	25,83	2,59	2,07
C06-II.2	8	0,65	0,28	0,85	4,59	0,00	1,64
C10-II.2	10	0,23	1,80	1,80	29,92	1,03	1,47
C11-II.2	15	1,89	479,11	1,65	15,85	1,35	0,95
C12-II.2	5	3,05	126,34	16,75	-	5,72	104,12
D04-II.2	12	0,00	0,19	0,00	1,42	0,00	0,06
D05-II.2	4	0,00	0,07	0,32	6,46	0,32	0,33
D06-II.2	8	0,00	0,12	0,00	1,08	0,00	0,05
D10-II.2	10	0,68	155,14	1,56	97,97	1,56	4,91
D11-II.2	15	0,22	7,46	0,00	21,79	0,00	0,14
D12-II.2	5	0,73	29,22	5,92	-	3,04	114,41
E04-II.2	12	0,18	0,66	0,00	6,66	0,00	1,42
E05-II.2	4	0,00	0,06	0,81	53,74	0,65	1,76
E06-II.2	8	0,19	0,32	0,27	4,27	0,08	0,30
E10-II.2	10	0,06	2,05	0,50	73,35	0,39	1,89
E11-II.2	15	0,51	274,17	0,48	19,84	0,35	1,51
E12-II.2	5	0,83	98,84	3,54	-	1,68	201,83

Tabela 5.16: Comparação de desempenho do algoritmo *branch-and-price*, usando a Definição 4.3, com o algoritmo *branch-and-cut* apresentado em Rodríguez-Martín et al. (2016b), em suas versões básica e completa.

CAPÍTULO

6

Contribuições e Conclusões

Este trabalho apresentou um novo modelo de programação linear inteira para o RRP e propôs um algoritmo *branch-and-price* para resolvê-lo. Diferentes algoritmos foram propostos para resolver o problema de *pricing* que surge neste tipo de algoritmo. Heurísticas de *pricing* e heurísticas primais também foram propostas para melhorar o desempenho do algoritmo e uma análise foi apresentada com base em experimentos empíricos realizados em instâncias da literatura.

Os resultados deste trabalho foram apresentados em dois eventos da área de otimização:

- V Encontro de Teoria da Computação - ETC'20 no XL SBC;
- XI Latin and American Algorithms, Graphs and Optimization Symposium - LAGOS'21.

O trabalho apresentado no LAGOS será publicado no *Procedia Computer Science*.

Como extensões do trabalho, acreditamos que a inclusão de desigualdades válidas neste modelo possa obter limitantes duais melhores, melhorando o seu desempenho, assim como documentado na literatura em outras variantes do VRP.

Referências Bibliográficas

Alumur, S. A., Campbell, J. F., Contreras, I., Kara, B. Y., Marianov, V., e O'Kelly, M. E. (2021). Perspectives on modeling hub location problems. *European Journal of Operational Research*, 291(1):1–17. Citado na página 3.

Baldacci, R., Dell'Amico, M., e González, J. S. (2007). The capacitated m-ring-star problem. *Operations Research*, 55(6):1147–1162. Citado na página 4.

Baldacci, R., Mingozzi, A., e Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283. Citado na página 10.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., e Vance, P. (1970). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46. Citado na página 9.

Cplex (2015). *IBM ILOG CPLEX Optimization Studio CP Optimizer User's Manual, Version 12 Release 6*. IBM ILOG. Citado na página 31.

Dantzig, G. B., Orden, A., Wolfe, P., et al. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5(2):183–195. Citado na página 6.

Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91. Citado na página 4.

Feo, T. A. e Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71. Citado na página 12.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., e Werneck, R. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511. Citado na página 29.

Gamrath, G., Fischer, T., Gally, T., Gleixner, A. M., Hendel, G., Koch, T., Maher, S. J., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Vigerske, S., Weninger, D., Winkler, M., Witt, J. T., e Witzig, J. (2016). The SCIP Optimization Suite 3.2. Technical report, Optimization Online. Citado na página 31.

Hill, A. e Voß, S. (2016). Optimal capacitated ring trees. *EURO Journal on Computational Optimization*, 4(2):137–166. Citado nas páginas 4 e 32.

Hoshino, E. A. e De Souza, C. C. (2012). A branch-and-cut-and-price approach for the capacitated m-ring–star problem. *Discrete Applied Mathematics*, 160(18):2728–2741. Citado na página 4.

Hoshino, E. A. e Hill, A. (2014). Column generation approach for the capacitated ring tree problem. In *VIII Workshop on Applied Combinatorial Optimization (ALIO/EURO)*, Universidad de la República, Uruguay. Citado na página 4.

Lewis, C. (2008). Linear programming: Theory and applications. *Whitman College Mathematics Department*. Citado na página 5.

Lübbecke, M. E. (2010). Column generation. In *Wiley encyclopedia of operations research and management science. Wiley, New York*, páginas 1–14. Citado nas páginas 7 e 8.

Miller, C. E., Tucker, A. W., e Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329. Citado na página 9.

Rodríguez-Martín, I., Salazar-González, J.-J., e Yaman, H. (2016a). A branch-and-cut algorithm for two-level survivable network design problems. *Computers & Operations Research*, 67:102–112. Citado nas páginas xiii, 3, 16, 17, 21, 32, 43, e 45.

Rodríguez-Martín, I., Salazar-González, J.-J., e Yaman, H. (2016b). The ring/κ-rings network design problem: Model and branch-and-cut algorithm. *Networks*, 68(2):130–140. Citado nas páginas xiii, 3, 16, 17, 22, 32, 45, e 46.

Thomadsen, T. e Stidsen, T. (2005). Hierarchical ring network design using branch-and-price. *Telecommunication Systems*, 29(1):61–76. Citado na página 3.

Wolsey, L. A. (1998). *Integer programming*. Wiley, New York. Citado nas páginas 6, 8, e 9.

Apêndice

As Tabelas 1, 2 e 3 mostram os dados de todas as instâncias do Grupo A, e das classes I e II do Grupo B, respectivamente.

Tabela 1: Dados das instâncias do Grupo A.

instância	$\|V\|$	$\|E\|$	Q	k
A01	13	78	5	3
A02	13	78	4	4
A03	13	78	3	5
A04	19	171	7	3
A05	19	171	5	4
A06	19	171	4	5
A07	26	325	10	3
A08	26	325	7	4
A09	26	325	6	5
A16	38	703	14	3
A17	38	703	11	4
A18	38	703	9	5
A19	51	1275	19	3
A20	51	1275	14	4
A21	51	1275	12	5
A22	19	171	7	3
A23	19	171	5	4
A24	19	171	4	5
A25	38	703	14	3
A26	38	703	11	4
A27	38	703	9	5
A28	57	1596	21	3
A29	57	1596	16	4

continuação na próxima página...

instância	$\|V\|$	$\|E\|$	Q	k
A30	57	1596	13	5
A31	76	2850	28	3
A32	76	2850	21	4
A33	76	2850	17	5
A34	26	325	10	3
A35	26	325	7	4
A36	26	325	6	5
A37	51	1275	19	3
A38	51	1275	14	4
A39	51	1275	12	5
A40	76	2850	28	3
A41	76	2850	21	4
A42	76	2850	17	5
A43	101	5050	38	3
A44	101	5050	28	4
A45	101	5050	23	5

Tabela 2: Dados das instâncias do Grupo B - classe I.

$k=1$	$k=2$	classe	f_i	$\|V\|$	$\|E\|$	Q	k
C01-I.1	C04-I.2	I	[1,500]	15	105	12	1
C02-I.1	C05-I.2	I	[1,500]	15	105	4	1
C03-I.1	C06-I.2	I	[1,500]	15	105	8	1
C07-I.1	C10-I.2	I	[1,500]	20	190	10	1
C08-I.1	C11-I.2	I	[1,500]	20	190	15	1
C09-I.1	C12-I.2	I	[1,500]	20	190	5	1
C13-I.1	C16-I.2	I	[1,500]	30	435	15	1
C14-I.1	C17-I.2	I	[1,500]	30	435	23	1
C15-I.1	C18-I.2	I	[1,500]	30	435	8	1
C19-I.1	C22-I.2	I	[1,500]	40	780	10	1
C20-I.1	C23-I.2	I	[1,500]	40	780	20	1
C21-I.1	C24-I.2	I	[1,500]	40	780	30	1
D01-I.1	D04-I.2	I	[500,1000]	15	105	12	1
D02-I.1	D05-I.2	I	[500,1000]	15	105	4	1
D03-I.1	D06-I.2	I	[500,1000]	15	105	8	1
D07-I.1	D10-I.2	I	[500,1000]	20	190	10	1
D08-I.1	D11-I.2	I	[500,1000]	20	190	15	1
D09-I.1	D12-I.2	I	[500,1000]	20	190	5	1

continuação na próxima página...

$k=1$	$k=2$	classe	f_i	$\|V\|$	$\|E\|$	Q	k
D13-I.1	D16-I.2	I	[500,1000]	30	435	15	1
D14-I.1	D17-I.2	I	[500,1000]	30	435	23	1
D15-I.1	D18-I.2	I	[500,1000]	30	435	8	1
D19-I.1	D22-I.2	I	[500,1000]	40	780	10	1
D20-I.1	D23-I.2	I	[500,1000]	40	780	20	1
D21-I.1	D24-I.2	I	[500,1000]	40	780	30	1
E01-I.1	E04-I.2	I	[1200,1700]	15	105	12	1
E02-I.1	E05-I.2	I	[1200,1700]	15	105	4	1
E03-I.1	E06-I.2	I	[1200,1700]	15	105	8	1
E07-I.1	E10-I.2	I	[1200,1700]	20	190	10	1
E08-I.1	E11-I.2	I	[1200,1700]	20	190	15	1
E09-I.1	E12-I.2	I	[1200,1700]	20	190	5	1
E13-I.1	E16-I.2	I	[1200,1700]	30	435	15	1
E14-I.1	E17-I.2	I	[1200,1700]	30	435	23	1
E15-I.1	E18-I.2	I	[1200,1700]	30	435	8	1
E19-I.1	E22-I.2	I	[1200,1700]	40	780	10	1
E20-I.1	E23-I.2	I	[1200,1700]	40	780	20	1
E21-I.1	E24-I.2	I	[1200,1700]	40	780	30	1

Tabela 3: Dados das instâncias do Grupo B - classe II.

$k=1$	$k=2$	classe	f_i	$\|V\|$	$\|E\|$	Q	k
C01-II.1	C04-II.2	II	[1,500]	15	105	12	1
C02-II.1	C05-II.2	II	[1,500]	15	105	4	1
C03-II.1	C06-II.2	II	[1,500]	15	105	8	1
C07-II.1	C10-II.2	II	[1,500]	20	190	10	1
C08-II.1	C11-II.2	II	[1,500]	20	190	15	1
C09-II.1	C12-II.2	II	[1,500]	20	190	5	1
C13-II.1	C16-II.2	II	[1,500]	30	435	15	1
C14-II.1	C17-II.2	II	[1,500]	30	435	23	1
C15-II.1	C18-II.2	II	[1,500]	30	435	8	1
C19-II.1	C22-II.2	II	[1,500]	40	780	10	1
C20-II.1	C23-II.2	II	[1,500]	40	780	20	1
C21-II.1	C24-II.2	II	[1,500]	40	780	30	1
D01-II.1	D04-II.2	II	[500,1000]	15	105	12	1
D02-II.1	D05-II.2	II	[500,1000]	15	105	4	1
D03-II.1	D06-II.2	II	[500,1000]	15	105	8	1
D07-II.1	D10-II.2	II	[500,1000]	20	190	10	1

continuação na próxima página...

$k=1$	$k=2$	classe	f_i	$\|V\|$	$\|E\|$	Q	k
D08-II.1	D11-II.2	II	[500,1000]	20	190	15	1
D09-II.1	D12-II.2	II	[500,1000]	20	190	5	1
D13-II.1	D16-II.2	II	[500,1000]	30	435	15	1
D14-II.1	D17-II.2	II	[500,1000]	30	435	23	1
D15-II.1	D18-II.2	II	[500,1000]	30	435	8	1
D19-II.1	D22-II.2	II	[500,1000]	40	780	10	1
D20-II.1	D23-II.2	II	[500,1000]	40	780	20	1
D21-II.1	D24-II.2	II	[500,1000]	40	780	30	1
E01-II.1	E04-II.2	II	[1200,1700]	15	105	12	1
E02-II.1	E05-II.2	II	[1200,1700]	15	105	4	1
E03-II.1	E06-II.2	II	[1200,1700]	15	105	8	1
E07-II.1	E10-II.2	II	[1200,1700]	20	190	10	1
E08-II.1	E11-II.2	II	[1200,1700]	20	190	15	1
E09-II.1	E12-II.2	II	[1200,1700]	20	190	5	1
E13-II.1	E16-II.2	II	[1200,1700]	30	435	15	1
E14-II.1	E17-II.2	II	[1200,1700]	30	435	23	1
E15-II.1	E18-II.2	II	[1200,1700]	30	435	8	1
E19-II.1	E22-II.2	II	[1200,1700]	40	780	10	1
E20-II.1	E23-II.2	II	[1200,1700]	40	780	20	1
E21-II.1	E24-II.2	II	[1200,1700]	40	780	30	1

As Tabelas 4, 5 e 6 apresentam os resultados obtidos pelo algoritmo *branch-and-price*, usando-se os parâmetros calibrados e a melhor configuração de implementação nas instâncias do Grupo A e nas classes I e II do Grupo B, respectivamente. O tempo limite de execução foi fixado em 900 segundos. O símbolo $-$ é usado quando o tempo limite é atingido ou quando o algoritmo terminou sem encontrar um limitante dual no nó raiz. O símbolo $*$ indica que nenhum *gap* de dualidade está disponível devido ao limite de tempo. As instâncias assinaladas com "**" são inviáveis na Definição 4.1 em razão dos limites Q e k.

Tabela 4: Resultados para as instâncias do Grupo A

instância	lb	gap	tempo	nós
A01	144,00	0,00	1,21	87
A02	158,00	0,00	1,22	175
A03	168,00	0,00	0,13	28
A04	174,00	0,00	70,08	1057
A05	188,00	0,00	29,59	990

continuação na próxima página...

instância	lb	gap	tempo	nós
A06	202,00	0,00	9,20	473
A07	168,27	15,88	-	738
A08	183,25	8,59	-	2295
A09	194,78	8,84	-	3478
A16	184,93	27,62	-	392
A17	190,86	33,08	-	971
A18	198,75	41,88	-	874
A19	221,07	25,75	-	4
A20	225,90	46,53	-	81
A21	229,13	47,08	-	204
A22	163,00	0	11,03	218
A23	189,00	0	27,64	1080
A24	207,00	0	4,70	276
A25	197,44	20,04	-	312
A26	201,21	40,65	-	349
A27	207,61	52,69	-	377
A28	-	*	-	1
A29	222,72	80,49	-	6
A30	226,59	73	-	31
A31	-	*	-	1
A32	-	*	-	1
A33	-	*	-	1
A34	193,36	18,43	-	1548
A35	202,77	27,73	-	1128
A36	212,91	13,66	-	2875
A37	237,06	74,64	-	2
A38	241,61	70,11	-	81
A39	247,46	59,62	-	217
A40	-	*	-	1
A41	-	*	-	1
A42	-	*	-	1
A43	-	*	-	1
A44	-	*	-	1
A45	-	*	-	1

Tabela 5: Resultados para as instâncias da classe I do Grupo *B*

instância	lb	gap	tempo	nós
C01-I.1	602,00	0,00	18,37	41
C02-I.1	**	**	**	**
C03-I.1	654,00	0,00	18,92	153
C04-I.2	601,00	0,00	23,39	60
C05-I.2	812,00	0,00	0,50	41
C06-I.2	605,00	0,00	8,20	75
C07-I.1	609,00	0,00	54,59	176
C08-I.1	477,00	0,00	18,35	3
C09-I.1	1113,00	0,00	2,61	57
C10-I.2	517,00	0,00	36,39	172
C11-I.2	477,00	0,00	57,50	11
C12-I.2	721,00	0,00	0,28	5
C13-I.1	418,12	3,80	-	96
C14-I.1	347,78	170,29	-	3
C15-I.1	680,87	2,08	-	2315
C16-I.2	388,02	10,56	-	75
C17-I.2	339,00	68,44	-	7
C18-I.2	461,71	15,87	-	1272
C19-I.1	586,92	21,65	-	385
C20-I.1	379,43	226,81	-	6
C21-I.1	-	*	-	1
C22-I.2	438,51	15,85	-	299
C23-I.2	331,89	204,92	-	5
C24-I.2	-	*	-	1
D01-I.1	1125,00	0,00	31,4	35
D02-I.1	**	**	**	**
D03-I.1	1276,00	0,00	2,93	29
D04-I.2	1125,00	0,00	34,70	65
D05-I.2	1980,00	0,00	1,11	131
D06-I.2	1171,00	0,00	23,84	204
D07-I.1	1685,00	0,00	720,72	1045
D08-I.1	1025,00	0,00	199,07	34
D09-I.1	3322,00	0,00	10,67	383
D10-I.2	1031,00	0,00	38,22	32
D11-I.2	1006,81	5,78	-	248
D12-I.2	1838,00	0,00	4,71	115

continuação na próxima página...

instância	lb	gap	tempo	nós
D13-I.1	977,39	68,10	-	106
D14-I.1	721,62	155,81	-	2
D15-I.1	2267,33	34,25	-	1441
D16-I.2	729,49	51,34	-	51
D17-I.2	585,67	150,48	-	6
D18-I.2	1358,65	18,57	-	1195
D19-I.1	2284,6	18,88	-	298
D20-I.1	1020,73	123,57	-	12
D21-I.1	774,00	174,29	-	2
D22-I.2	1401,70	7,44	-	276
D23-I.2	725,03	82,75	-	7
D24-I.2	-	*	-	1
E01-I.1	1802,00	0,00	38,28	53
E02-I.1	**	**	*	**
E03-I.1	1854,00	0,00	0,85	5
E04-I.2	1801,00	0,00	35,85	58
E05-I.2	3212,00	0,00	1,75	213
E06-I.2	1805,00	0,00	11,45	93
E07-I.1	3009,00	0,00	189,04	489
E08-I.1	1677,00	0,00	293,42	41
E09-I.1	5913,00	0,00	8,33	219
E10-I.2	1717,00	0,00	198,08	337
E11-I.2	1677,00	0,00	343,80	45
E12-I.2	3121,00	0,00	1,36	25
E13-I.1	1748,10	86,77	-	72
E14-I.1	945,08	137,86	-	2
E15-I.1	4410,03	24,74	-	1781
E16-I.2	1204,95	34,53	-	63
E17-I.2	758,24	185,00	-	6
E18-I.2	2859,95	2,66	-	1562
E19-I.1	4610,09	22,12	-	408
E20-I.1	1726,30	143,93	-	27
E21-I.1	1006,67	189,67	-	2
E22-I.2	2471,41	24,10	-	323
E23-I.2	1098,49	98,82	-	7
E24-I.2	-	*	-	1

Tabela 6: Resultados para as instâncias da classe II do Grupo *B*.

instância	lb	gap	tempo	nós
C01-II.1	785,00	0,00	4,42	5
C02-II.1	**	**	**	**
C03-II.1	1111,00	0,00	383,68	3401
C04-II.2	785,00	0,00	709,56	728
C05-II.2	1255,00	0,00	2,14	210
C06-II.2	860,00	0,00	140,55	987
C07-II.1	876,62	3,47	-	2165
C08-II.1	779,00	0,00	109,39	77
C09-II.1	1467,00	0,00	8,19	147
C10-II.2	695,00	0,00	108,18	339
C11-II.2	617,00	0,00	81,22	61
C12-II.2	1018,00	0,00	1,18	24
C13-II.1	842,57	38,86	-	159
C14-II.1	729,87	65,10	-	30
C15-II.1	1136,18	25,68	-	819
C16-II.2	798,00	0,00	681,20	125
C17-II.2	668,99	24,52	-	34
C18-II.2	977,62	26,74	-	1343
C19-II.1	1235,72	55,38	-	150
C20-II.1	859,52	49,97	-	34
C21-II.1	735,13	139,01	-	4
C22-II.2	986,48	51,45	-	186
C23-II.2	759,89	91,87	-	33
C24-II.2	669,62	136,55	-	4
D01-II.1	1273,00	0	17,02	38
D02-II.1	**	**	**	**
D03-II.1	1571,00	0,00	15,97	118
D04-II.2	1273,00	0,00	208,24	413
D05-II.2	2120,00	0,00	0,42	25
D06-II.2	1303,00	0,00	5,07	41
D07-II.1	1985,00	0,00	190,54	601
D08-II.1	1309,00	0,00	173,08	69
D09-II.1	3790,00	0,00	56,27	999
D10-II.2	1211,00	0,00	50,23	184
D11-II.2	1133,00	0,00	248,00	103
D12-II.2	2160,00	0,00	7,18	210

continuação na próxima página...

instância	lb	gap	tempo	nós
D13-II.1	1701,09	32,15	-	177
D14-II.1	1286,17	59,39	-	32
D15-II.1	2778,43	20,25	-	2290
D16-II.2	1263,04	12,19	-	143
D17-II.2	975,28	55,54	-	25
D18-II.2	1837,73	27,44	-	547
D19-II.1	2905,92	41,13	-	254
D20-II.1	1718,55	17,48	-	19
D21-II.1	1326,31	62,86	-	3
D22-II.2	1934,73	24,62	-	301
D23-II.2	1206,34	12,49	-	41
D24-II.2	968,00	39,26	-	4
E01-II.1	1985,00	0,00	31,17	31
E02-II.1	**	**	**	**
E03-II.1	2340,00	0,00	7,86	57
E04-II.2	1985,00	0,00	281,65	221
E05-II.2	3655,00	0,00	2,99	305
E06-II.2	2060,00	0,00	111,46	611
E07-II.1	3279,67	0,56	-	2032
E08-II.1	1979,00	0,00	312,68	84
E09-II.1	6267,00	0,00	55,65	1245
E10-II.2	1895,00	0,00	233,91	430
E11-II.2	1817,00	0,00	268,56	99
E12-II.2	3418,00	0,00	3,15	57
E13-II.1	2450,97	38,39	-	160
E14-II.1	1789,51	82,56	-	32
E15-II.1	4977,94	28,51	-	466
E16-II.2	1836,62	7,10	-	148
E17-II.2	1386,88	46,30	-	32
E18-II.2	3245,58	7,72	-	1565
E19-II.1	5324,44	29,35	-	457
E20-II.1	2506,91	64,51	-	25
E21-II.1	1962,69	74,00	-	2
E22-II.2	3127,10	24,97	-	401
E23-II.2	1817,14	38,18	-	17
E24-II.2	1397,87	148,23	-	2