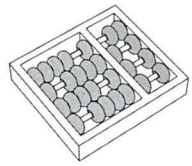


Mauro C. Lopes

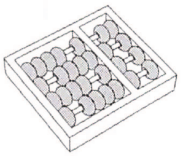
"Um problema integrado de localização e roteamento com transporte entre concentradores e relação de muitos-para-muitos"

CAMPINAS
2014

ERRATA

Onde se lê: Mauro C. Lopes
Leia-se: Mauro Cardoso Lopes

Prof. Paulo Lício de Geus
Coord. de Pós-Graduação
Instituto de Computação - Unicamp
Matrícula 10.326-8



Universidade Estadual de Campinas
Instituto de Computação

Mauro C. Lopes

"Um problema integrado de localização e roteamento com transporte entre concentradores e relação de muitos-para-muitos"

Orientador(a): **Prof. Dr. Flávio Keidi Miyazawa**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Ciência da Computação.

Este exemplar corresponde à versão final da Dissertação defendida por Mauro C. Lopes, sob orientação de Prof. Dr. Flávio Keidi Miyazawa.

Assinatura do Orientador(a)

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Maria Fabiana Bezerra Muller - CRB 8/6162

L881p

Lopes, Mauro Cardoso, 1988-
Um problema integrado de localização e roteamento com transporte entre concentradores e relação de muitos-para-muitos / Mauro Cardoso Lopes. – Campinas, SP : [s.n.], 2014.

Orientador: Flávio Keidi Miyazawa.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Algoritmo Branch and Cut. 2. Busca local. 3. Problema de roteamento de veículos. 4. Otimização combinatória. I. Miyazawa, Flávio Keidi,1970-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Many-to-many location-routing with inter-hub transport
Palavras-chave em inglês:
Branch-and-Cut algorithms
Local search
Vehicle routing problem
Combinatorial optimization
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Flávio Keidi Miyazawa [Orientador]
Eduardo Candido Xavier
Thiago Alves de Queiroz
Data de defesa: 27-06-2014
Programa de Pós-Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Defesa de Dissertação de Mestrado em Ciência da Computação, apresentada pelo(a) Mestrando(a) **Mauro Cardoso Lopes**, aprovado(a) em **27 de junho de 2014**, pela Banca examinadora composta pelos Professores Doutores:

Prof(a). Dr(a). Thiago Alves de Queiroz
Titular

Prof(a). Dr(a). Eduardo Candido Xavier
Titular

Prof(a). Dr(a). Flávio Keidi Miyazawa
Presidente

Instituto de Computação
Universidade Estadual de Campinas

Um problema integrado de localização e roteamento com transporte entre concentradores e relação de muitos-para-muitos

Mauro C. Lopes[1]

27 de junho de 2014

Banca Examinadora:

- Prof. Dr. Flávio Keidi Miyazawa (Orientador)
- Prof. Dr. Thiago Alves de Queiroz
  Departamento de Matemática - UFG
- Prof. Dr. Eduardo Candido Xavier
  Instituto de Computação - UNICAMP
- Prof. Dr. Luis Augusto Angelotti Meira (Suplente)
  Faculdade de Tecnologia - UNICAMP
- Prof. Dr. Fábio Luiz Usberti (Suplente)
  Instituto de Computação - UNICAMP

[1]Bolsa CAPES 22133/2012

Resumo

Investigamos uma variante do problema de localização e roteamento com relação de muitos-para-muitos concentradores que consiste em particionar o conjunto de vértices de um grafo em ciclos contendo exatamente um concentrador cada e determinar um ciclo adicional interligando todos os concentradores. Qualquer vértice do grafo pode ser um concentrador; faz parte do problema determinar quais vértices devem ser concentradores. Esse problema tem aplicações práticas relevantes em áreas como transporte urbano e redes de computadores.

Desenvolvemos uma heurística baseada em busca local com operações de inserção, remoção e troca de vértices. Soluções iniciais são geradas de maneira aleatória, e suas vizinhanças são exploradas a fim de obter melhores soluções. Além disso, elaboramos um algoritmo exato com estrutura de *branch-and-cut* para a formulação em Programação Linear Inteira proposta. Restrições de capacidade e eliminação de caminhos são adicionadas como planos de corte, com algoritmos de separação baseados em árvores de corte mínimo e nas componentes conexas de um grafo suporte.

Diversos experimentos computacionais mostram a capacidade de resolução do algoritmo exato para instâncias pequenas e da heurística para instâncias pequenas e médias. São comparados também os desempenhos para outras variantes do problema.

Palavras-chave: Localização e roteamento; Localização muitos-para-muitos; Algoritmo *Branch and Cut*; Busca local; Problema de roteamento de veículos

Abstract

We investigate a variant of the many-to-many hub location-routing problem which consists in partitioning the set of vertices of a graph into cycles containing exactly one hub each and determining an extra cycle interconnecting all hubs. Any vertex of the graph can be a hub; it is part of the problem to determine which vertices should be hubs. This problem has relevant practical applications in areas such as urban transportation and computer networks.

A local search based heuristic that considers add/remove and swap operations is developed. Initial solutions can be generated at random, and their neighborhoods are explored in order to get better solutions. Also a branch-and-cut approach that solves an integer formulation is investigated. Capacity and path elimination constraints are added in a cutting plane way, so the separation algorithms are based on the computation of min-cut trees and in the connected components of a support graph.

Many computational experiments over several instances adapted from literature show the problem-solving capability of the exact algorithm for small instances and of the heuristic for small to medium-sized instances. We also compare the performance of other variants of the problem.

Keywords: Location-routing; Many-to-many hub location; Branch and Cut; Local search; Vehicle routing problem

Agradecimentos

Antes de tudo, agradeço imensamente à minha família, que tem me apoiado incondicionalmente desde sempre. Sem eles, nada disso seria possível.

Muito obrigado a todo o pessoal que trabalha no Instituto de Computação da Unicamp, professores e funcionários, por oferecerem um curso de tamanha excelência.

Agradeço também a todos os colegas do LOCo que me ajudaram ao longo do mestrado, fosse com discussões teóricas, ou com momentos de descontração na hora das refeições, ou mesmo ajudando a resolver todo tipo de problema em discussões *online* madrugada adentro. Em especial, quero mencionar meus amigos Carlos, Evandro e Thiago, que, cada um à sua maneira, contribuíram muito para eu obter sucesso.

Como não poderia deixar de ser, agradeço ao Professor Flávio, que muito me ensinou ao longo de inúmeras discussões e reuniões de uma hora que duravam duas ou três.

Por fim, agradeço à Capes pelo apoio financeiro que me deu suporte para me concentrar nos estudos.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Otimização Combinatória é uma área da Teoria da Computação que trata de problemas em que se deve minimizar ou maximizar uma função sob um conjunto de restrições, em um domínio finito. Nessa área, encontram-se os problemas de roteamento, que consistem em encontrar, em um grafo, trajetos com determinadas propriedades minimizando a quantidade de arestas ou soma de seus custos, que podem representar distância percorrida, tempo ou combustível gasto, por exemplo. Eles estão entre os mais estudados da área. Isso se deve ao fato de tais problemas terem diversas aplicações práticas na indústria e em serviços, como transporte, logística, circuitos elétricos e redes de computadores.

Nesse contexto, estão inseridos dois problemas computacionais famosos: o Problema do Caixeiro Viajante e o Problema de Roteamento de Veículos. O primeiro consiste em visitar exatamente uma vez cada vértice de um grafo e retornar ao local de origem, minimizando a distância percorrida. O segundo, em encontrar trajetos, todos originados e terminados em um mesmo ponto e que, combinados, passem por cada vértice exatamente uma vez, com o menor comprimento ou custo possível.

Esses problemas são importantes por alguns motivos. Primeiro, porque são de enunciado relativamente simples. Segundo, sendo $\mathcal{NP}$-difíceis, não existem métodos exatos de resolvê-los eficientemente. Terceiro, porque possuem muitas aplicações em áreas como transportes, eletrônica e redes de computadores. E, por fim, porque servem de base para inúmeros outros problemas mais específicos.

Em transporte de carga, concentradores são instalações especiais que servem como pontos de conexão entre origens e destinos, nos quais os produtos oriundos das origens são consolidados e redirecionados para seus respectivos destinos. Essa estrutura de rede tem seus benefícios, por exemplo quando um destino recebe produtos de diversas origens (caso muitos-para-um), ou uma origem provém um produto para diferentes destinos (caso um-para-muitos), ou ainda quando muitas origens fornecem produtos para vários destinos (caso

muitos-para-muitos). Nessas situações, as rotas que vão até os destinos são utilizadas para transportar produtos que serão consolidados e reenviados por rotas interligadoras (com ligações entre concentradores) para concentradores finais, e depois encaminhadas por esses concentradores finais até os destinatários que eles atendem. Por isso, recebe o nome de Problema integrado de Localização e Roteamento com transporte entre concentradores e relação de Muitos-para-Muitos – LRCMM. A Figura 1.1 ilustra o caso em que os produtos dos clientes n_1, n_2 e n_3 são enviados (uma seta indica o fluxo entre vértices) para seus respectivos concentradores e, destes, usando a rota interligadora, são transportados até o concentrador h_3, que atende o cliente n_4.

Neste trabalho, investigamos o caso muitos-para-muitos com as seguintes hipóteses: qualquer vértice é candidato a ser concentrador na rede; o número de concentradores necessários é dado na entrada; cada cliente é atendido por exatamente um concentrador e precisa estar em exatamente uma rota local, para que a coleta e a entrega ocorram simultaneamente; em cada concentrador começa e termina apenas uma rota local que sirva dois ou mais clientes (estudamos também, separadamente, o caso em que rotas locais podem atender um ou mesmo nenhum cliente); as capacidades dos concentradores são ilimitadas, mas qualquer rota local possui um número máximo de vértices, dado na entrada; os concentradores devem estar em uma rota circular interligadora. Há também um fator de desconto α utilizado na rota interligadora devido a transporte em massa – veja O'Kelly (1987). O objetivo é minimizar o total dado pela soma dos custos das arestas interligadoras, considerando o fator de desconto, mais os custos das arestas regulares.

O *LRCMM* possui diversas aplicações práticas. Uma área imediata de aplicação é a de transporte urbano. Considere que os ciclos regulares (ou rotas locais) são rotas de ônibus internas de cada bairro, enquanto que o ciclo de interligação é uma via expressa (seja uma avenida de alta velocidade com poucos pontos de parada, seja um outro meio de transporte como trens leves ou metrô, ou mesmo um corredor de ônibus, o que torna o sistema acessível a muito mais cidades). Deseja-se fazer um sistema de transporte, como o descrito, minimizando o custo de implantação das vias. Uma característica interessante é conectar quaisquer dois pontos da cidade usando no máximo dois transportes regulares e um expresso, o que torna a vida dos passageiros menos complicada.

Outra aplicação interessante do LRCMM ocorre em redes de computadores. Neste caso, temos várias redes locais interligadas por uma rede global, todas elas com topologia em anel. Podemos querer minimizar o custo de instalação ou a latência na transmissão de dados.

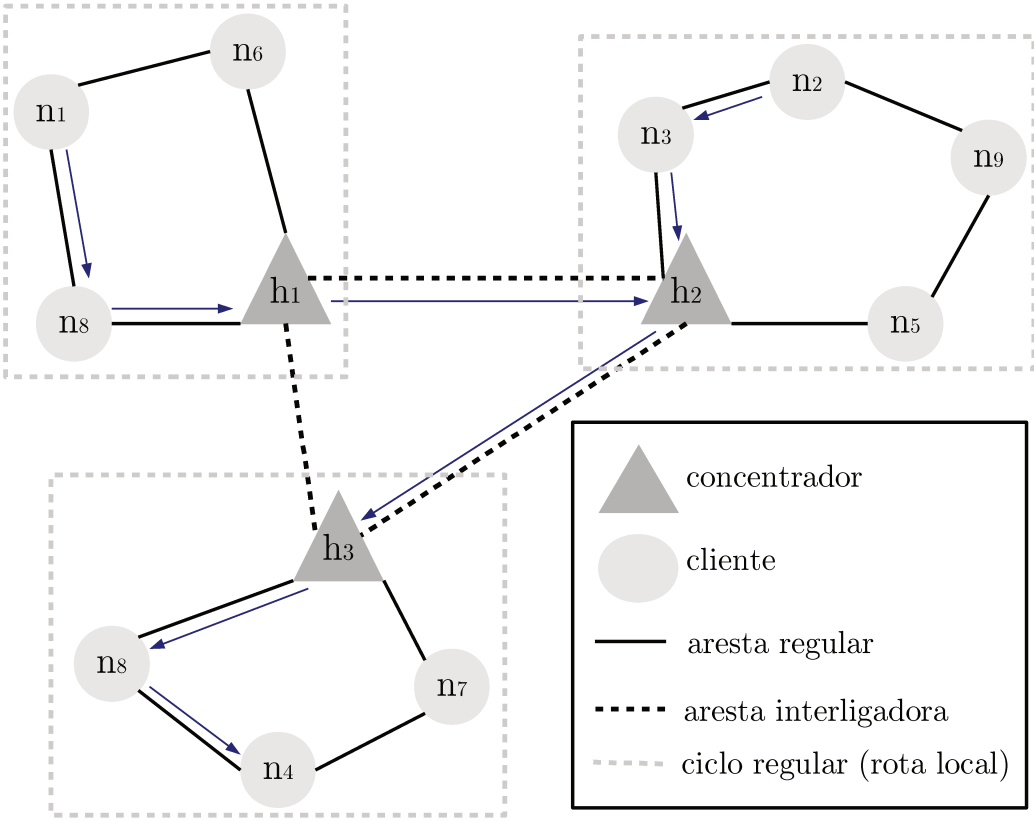


Figura 1.1: Ilustração de solução do LRCMM.

1.1 Atividades Realizadas

Neste trabalho, estudamos abordagens para o Problema integrado de Localização e Roteamento com transporte entre concentradores e relação de Muitos-para-Muitos (LRCMM), tanto de modo exato quanto heurístico. Para tanto:

- desenvolvemos e implementamos heurísticas de construção e de melhoria para usar em uma busca local;
- elaboramos uma formulação em Programação Linear Inteira;
- criamos e implementamos métodos de cortes específicos;
- pesquisamos e adaptamos métodos de cortes de problemas similares, e
- implementamos o método exato *Branch-and-cut* usando o *framework* do Gurobi.

1.2 Organização do Texto

O Capítulo 2 mostra uma visão geral de problemas similares que levaram à elaboração do LRCMM, bem como algumas técnicas frequentemente utilizadas para resolvê-los, algumas das quais foram implementadas neste trabalho.

O Capítulo 3 traz a maior parte do conteúdo do trabalho. Primeiro, define notações e nomenclatura usadas ao longo do texto. Depois, descreve o problema a ser estudado e define-o através de uma formulação em Programação Linear Inteira. Em seguida, apresenta cortes válidos que podem ser usados para fortalecer a formulação, com informações sobre a implementação de algoritmos de separação para encontrá-los. Descreve, ainda, a implementação de uma solução exata para o problema, utilizando a técnica de *Branch-and-cut*, incluindo as diversas rotinas de separação implementadas para encontrar desigualdades. Por fim, explica a abordagem heurística para a resolução do problema, apresenta o algoritmo utilizado e ilustra seu funcionamento.

O Capítulo 4 apresenta os resultados computacionais decorrentes da execução do algoritmo exato e da heurística, bem como a discussão dos resultados obtidos.

Finalmente, o Capítulo 5 faz as considerações finais e aponta possíveis caminhos para futuros desdobramentos deste estudo.

Capítulo 2

Revisão Bibliográfica

2.1 Técnicas de tratamento de problemas difíceis

Nesta seção, apresentamos conceitos básicos de Otimização Combinatória e mostramos como funcionam algumas técnicas exatas de resolução de problemas $\mathcal{NP}$-difíceis, tais como Programação Linear Inteira, Planos de Corte, *Branch-and-bound* e *Branch-and-cut*. Mostramos também o que são heurísticas e meta-heurísticas, o motivo de utilizá-las e alguns exemplos.

2.1.1 Otimização Combinatória

Podemos definir um problema de Otimização Combinatória como a tarefa de encontrar o ponto de máximo ou de mínimo global de uma função (*função objetivo*), sob um certo conjunto de restrições, e cujas variáveis pertencem a um domínio finito enumerável. Uma *solução viável* é uma atribuição de valores às variáveis de modo que todas as restrições sejam satisfeitas. O *valor da solução* é o valor da função objetivo para uma dada valoração. Por fim, uma *solução ótima* é uma solução viável cujo valor é melhor ou igual ao de qualquer outra solução viável.

Como consequência de possuir domínio finito enumerável, todo problema de Otimização Combinatória admite um algoritmo exato do tipo "força bruta" que enumera todas as soluções viáveis e encontra, se existir, alguma que seja ótima. Entretanto, o número de soluções viáveis pode crescer muito rapidamente com o tamanho da entrada. Por isso, na prática, é necessário encontrar meios para diminuir o espaço de busca, como algumas das técnicas a seguir.

2.1.2 Programação Linear Inteira

Programação Linear é uma forma de modelar problemas de otimização com restrições e função objetivo lineares (polinômios de primeiro grau). Se considerarmos também que as variáveis devem ser inteiras, trata-se de uma modelagem em Programação Linear Inteira

(PLI). É uma maneira muito útil de analisar problemas de Otimização Combinatória. Entretanto, o problema geral de PLI é $\mathcal{NP}$-difícil, ao contrário de Programação Linear, para o qual existem algoritmos de tempo polinomial conhecidos, conforme explicado por Dasgupta *et al.* (2008). Por isso, uma estratégia frequente é relaxar as restrições de integralidade das variáveis, permitindo valores fracionários. Nesse caso, o problema resultante é a *relaxação* do problema original. O valor ótimo dessa relaxação é um *limitante dual* (ou seja, limitante inferior para problemas de minimização, e superior para maximização). Em alguns problemas, os limitantes obtidos pela relaxação são muito bons, mas, em outros, são de praticamente nenhuma utilidade. O valor de qualquer ponto da região viável é um *limitante primal* para o problema.

2.1.3 *Branch-and-bound*

Como descrito por Doig *et al.* (1960), *Branch-and-bound* é uma estratégia geral de desenvolvimento de algoritmos exatos (que garantem encontrar uma solução ótima) para problemas de otimização combinatória. Trata-se basicamente de uma versão aprimorada do método *Backtracking* de enumerar todas as possíveis soluções, fixando o valor de uma variável a mais a cada ramificação da árvore de possibilidades. O método Branch-and-bound traz uma significativa vantagem: utiliza limitantes superiores e inferiores para eliminar grandes porções de soluções candidatas, sem precisar testar cada uma delas. Desse modo, sua execução costuma ser bem mais rápida do que a de um algoritmo completamente "força bruta", ainda que não possua garantias teóricas de ser mais rápido, no pior caso.

Em um problema de minimização, um limitante superior (primal) é dado pelo valor da melhor solução conhecida no momento. O limitante inferior (dual) para um dado nó é, naturalmente, menor ou igual ao valor de todas as soluções que podem vir a ser geradas por esse nó. Portanto, se o limitante superior global for menor que o limitante inferior de um dado nó, nenhuma solução gerada a partir do nó poderá resultar em uma solução melhor que a já conhecida. Logo, pode-se descartar esse nó e, com isso, evita-se ramificar a árvore e gerar mais possibilidades para serem testadas.

Uma das principais maneiras de encontrar limitantes duais em algoritmos de *Branch-and-bound* é utilizando uma formulação em Programação Linear Inteira para o problema em questão e relaxando suas restrições de integralidade, transformando em uma formulação de Programação Linear. Dessa forma, pode-se rapidamente resolver o problema relaxado e, por meio de bons limitantes, descartar o ramo atual.

2.1.4 Planos de Corte

Em problemas de otimização, uma técnica para reduzir o espaço de busca consiste em aplicar sucessivamente novas restrições na formulação matemática do problema. Essas restrições, chamadas *planos de corte*, são normalmente desigualdades lineares. Para serem corretos, os cortes não podem eliminar soluções viáveis. Para que a melhora do algoritmo seja substancial, é desejável que cada corte descarte uma parte grande do espaço de busca.

Em Programação Linear Inteira, isso consiste inicialmente em resolver a relaxação linear do problema. Se a solução ótima da relaxação possuir apenas variáveis com valores inteiros, tem-se uma solução viável do problema original e seu valor deve ser comparado ao limitante primal para possivelmente melhorá-lo. Caso contrário, deve-se *separar* a solução atual, isto é, eliminá-la do espaço de soluções viáveis por meio de uma nova restrição linear. Essa restrição não pode remover soluções viáveis do problema original, caso contrário, haveria risco de descartar soluções melhores, inclusive as ótimas. Este processo é repetido até que a solução da relaxação do problema satisfaça todas as restrições do problema original, incluindo as de integralidade. Nesse ponto, tem-se uma solução ótima do problema original. Mais detalhes sobre o assunto são dados por Mitchell (2002).

2.1.5 *Branch-and-cut*

A técnica de *Branch-and-cut* combina ideias de *Branch-and-bound* e de planos de corte. É usada para resolver problemas de Programação Linear Inteira. Baseada no *Branch-and-bound*, encontra uma solução ótima para a relaxação linear do problema, isto é, uma solução que satisfaz todas as restrições do problema, exceto possivelmente as restrições de integralidade das variáveis. Se esse valor for pior – ou igual – que o da melhor solução viável conhecida até o momento, esse nó pode ser descartado, acelerando o processo, já que não levará a uma solução viável com valor melhor. Senão, um ou mais planos de corte (restrições) são adicionados ao modelo. Essas restrições são violadas pela solução ótima fracionária atual, mas não pelas soluções inteiras viáveis. Pode ocorrer de não conhecermos um corte para adicionar ao modelo, embora a solução atual seja fracionária.

Nesse caso, particionamos o espaço de busca em dois, de modo a descartar a solução da relaxação atual, sem descartar soluções inteiras, o que é conhecido como ramificar (*branch*) a árvore de enumeração. Repete-se o processo até não mais existirem tais restrições que possam ser adicionadas. Se a execução do algoritmo chegar a esse ponto sem ter podado o ramo, então temos uma solução viável inteira (pois não há mais restrições que separem a solução da relaxação linear) com valor melhor que o atual melhor (já que o ramo atual não foi podado) e, portanto, podemos atualizar o valor do limitante primal. O processo continua até

que não existam mais nós a serem explorados, o que significa que a melhor solução conhecida até o momento é ótima. Note que todas as soluções viáveis, ou seja, todos os nós da árvore de possibilidades foram testados, direta ou indiretamente. É devido a essa garantia que o algoritmo é dito exato, já que prova que uma solução viável é ótima.

2.1.6 Heurísticas

Em muitos problemas, métodos exatos que os solucionem são desconhecidos, ou muito difíceis de implementar ou, mais frequentemente, têm execução extremamente lenta. Nesses casos, pode ser interessante recorrer a métodos computacionais conhecidos como heurísticas, com o intuito de obter rapidamente boas soluções viáveis para esses problemas.

São métodos que não garantem encontrar a solução ótima, nem oferecem garantias sobre a diferença (ou razão) entre o valor de uma solução viável encontrada e o valor das soluções ótimas. No caso geral, podem nem sequer garantir que encontrarão uma solução viável para o problema. Mesmo com todas essas incertezas, podem ser muito úteis ao serem usadas isoladamente quanto em conjunto com métodos exatos como *Branch-and-cut*.

Conforme explicado por Rothlauf (2011), podemos classificá-las em duas categorias: *heurísticas de construção*, que são aquelas usadas para encontrar uma solução inicial para um problema, e *heurísticas de melhoria*, que são usadas para encontrar uma solução melhor a partir de outra solução dada.

Meta-heurísticas são estratégias gerais (isto é, não específicas para o problema abordado), feitas para guiar o processo de busca por boas soluções em um problema de otimização. Existem diversas classes famosas de meta-heurísticas, dentre as quais podemos citar Busca Local, Busca Tabu, *Simulated Annealing*, além de Algoritmos Genéticos.

Algoritmos Genéticos são inspirados no processo biológico de seleção natural e foram introduzidos por Fraser (1957). Soluções viáveis (em geral, não ótimas) para o problema são formadas aleatoriamente e compõem a população inicial. Cada solução da população é um *indivíduo* com um certo valor (*fitness*) de função objetivo. A *elite* de uma dada população é um conjunto das melhores soluções (indivíduos com maiores *fitness* em problemas de maximização ou menores em problemas de minimização). Uma operação de *seleção* corresponde a escolher indivíduos da elite da população. A ideia de Algoritmos Genéticos consiste em iterativamente criar uma nova geração de indivíduos a partir da geração anterior, aplicando operações de seleção e de modificação. As modificações podem incluir *mutações* aleatórias na estrutura genética de indivíduos; *inversões* na ordem de subsequências de código genético; e *cruzamentos* entre dois indivíduos, de modo que cada novo indivíduo contenha subsequências dos genes de seus ancestrais. Dessa forma, espera-se que boas características sejam mantidas ao longo das gerações, tendendo a melhorar o valor da melhor solução conhecida.

Uma outra meta-heurística, que por sua vez é baseada em Algoritmos Genéticos, é a BRKGA (*Biased Random-Key Genetic Algorithms*). São necessárias duas rotinas auxiliares, o *codificador* e o *decodificador*, para converter soluções viáveis para vetores de números reais entre 0 e 1, e vice-versa. Para fazer um cruzamento, são sorteados dois indivíduos, sendo um deles da classe dos melhores indivíduos da população (a *elite*) e outro do grupo restante (ou da população geral). Como a elite é menor que a não-elite, um dado indivíduo da elite possui maior probabilidade de ser selecionado do que um dado indivíduo da não-elite. Além disso, é dado um parâmetro global, entre 0,5 e 1, representando a probabilidade de uma característica do filho ser herdada do pai pertencente à elite. Dessa forma, é esperado que as boas características tendam a permanecer nas gerações seguintes. Para mais detalhes, consulte Gonçalves e Resende (2011).

2.2 Publicações relacionadas

2.2.1 Problemas de roteamento

De modo geral, problemas de roteamento consistem em encontrar um conjunto de trajetos em um grafo de modo a atender demandas de clientes, dentro de certas restrições. Um trajeto é uma sequência de vértices em que cada par de vizinhos é ligado por uma aresta. Se ele começa e termina no mesmo vértice, é chamado de ciclo. Caso contrário, é um caminho. Esses conceitos serão definidos mais formalmente na seção 3.2.

Em 1735, o matemático suíço Leonhard Euler deu início ao estudo de Teoria dos Grafos ao resolver o Problema das Sete Pontes de Königsberg, o qual consiste em encontrar um caminho que passe exatamente uma vez em cada aresta de um determinado grafo – na versão original, o grafo foi construído a partir das pontes (arestas) ligando porções de terra (vértices) na cidade de Königsberg (Prússia), atual Kaliningrado (Rússia) – e que pode ser considerado o princípio do estudo de problemas de roteamento.

Já no Século XX, houve um grande aumento no interesse por essa área. Inúmeros artigos foram publicados abordando problemas em grafos, muitos deles relacionados a roteamento. O problema mais fundamental que aqui citamos é o clássico Problema do Ciclo Hamiltoniano (*Hamiltonian Cycle Problem* – HCP), nomeado em homenagem ao matemático irlandês W. R. Hamilton. Esse problema consiste em decidir, dado um grafo G, se existe um ciclo que visite todos os vértices exatamente uma vez – veja Thomason (1978). Em certas situações, a dificuldade não consiste em determinar a existência de tal ciclo (por exemplo, se o grafo for completo), mas sim em encontrar, dentre os ciclos que satisfazem essa condição, aquele cujo custo total (a soma dos custos de suas arestas) é mínimo. Nesse caso, trata-se do Problema do

Caixeiro Viajante (*Traveling Salesman Problem* – TSP), um problema que vem sendo muito estudado há décadas. O estudo dele originou diversas técnicas de abordagem de problemas, tanto gerais quanto específicas. Isso permitiu que fossem resolvidas instâncias muito maiores do que as de outros problemas $\mathcal{NP}$-difíceis. Atualmente, a maior instância já resolvida possui 85.900 vértices. Esse problema é tão importante que é tema central de diversos livros, tais como os escritos por Lawler (1985), Applegate *et al.* (2007) e Cook (2011), além de uma grande quantidade de artigos e outras publicações.

Enquanto o Problema do Ciclo Hamiltoniano e o Problema do Caixeiro Viajante são exemplos de problemas focados nos vértices de um grafo, existem também problemas relacionados às arestas. O Problema do Ciclo Euleriano é a versão análoga do Problema do Ciclo Hamiltoniano para arestas, ou seja, deve-se decidir se existe um ciclo que visite todas as arestas do grafo apenas uma vez; ao contrário daquele, este problema pode ser resolvido em tempo polinomial. Um problema similar de otimização é o Problema do Carteiro Chinês (*Chinese Postman Problem* – CPP), o qual consiste em encontrar o ciclo de menor custo que visite (uma ou mais vezes) todas as arestas de um grafo não-orientado. Em uma variante deste, o Problema do Carteiro Rural (*Rural Postman Problem* – RPP), precisa-se encontrar um ciclo hamiltoniano (isto é, que passe por todos os vértices) de custo mínimo que visite todas as arestas de um dado subconjunto das arestas do grafo.

Um outro problema especialmente conhecido é o Problema do Roteamento de Veículos (*Vehicle Routing Problem* – VRP), cujo objetivo é minimizar o custo total de um *conjunto de ciclos*, todos eles passando por um dado vértice especial (chamado *depósito*), de modo que cada um dos outros vértices do grafo (os *clientes*) pertença a exatamente um ciclo. Existem muitas variantes deste problema, incluindo versões com múltiplos depósitos (*Multi-Depot Vehicle Routing Problem* – MDVRP); com restrições de tempo na entrega do produto para cada cliente (*Vehicle Routing Problem with Time Windows* – VRPTW); com diferentes pontos de coleta e entrega para cada produto (*Vehicle Routing Problem with Pickup and Delivery* – VRPPD); e com capacidades de clientes a serem atendidos por cada veículo (*Capacitated Vehicle Routing Problem* – CVRP). Nesta última, cada cliente possui uma demanda por um determinado produto e, em cada ciclo, a soma das demandas de seus clientes não pode ultrapassar a capacidade máxima do veículo. Esta última é a versão clássica enunciada no trabalho seminal de Dantzig e Ramser (1959).

O Problema de Localização de Instalações (*Facility Location Problem*) consiste em determinar os melhores locais para abertura de instalações a fim de atender a demanda por um certo produto ou serviço. Por exemplo, uma rede de supermercados deve escolher onde construirá seus centros de distribuição de modo a armazenar e enviar os produtos para as lojas de maneira eficiente. Diante disso, problemas de localização e roteamento (*Location-*

Routing Problems) são aqueles que misturam os dois tipos de problema, geralmente exigindo a determinação da localização de depósitos e, em seguida, algum roteamento a partir deles. Esse é o caso do problema que estudamos.

2.2.2 Problemas relacionados ao LRCMM

Aqui apresentamos uma lista de problemas relacionados ao LRCMM, todos eles $\mathcal{NP}$-difíceis, em uma ordem tal que cada um seja baseado em seu antecessor, com algumas restrições adicionais. Todos referem-se a suas versões em grafos não-orientados.

Um problema bem menos famoso, e que pode ser considerado derivado do TSP e do VRP, é aquele em que há um conjunto dado de depósitos, cada um dos quais deve pertencer a um ciclo diferente, e os vértices dos ciclos devem formar uma partição do conjunto de vértices do grafo. Esse problema é denominado Problema do Caixeiro Viajante Múltiplo com Múltiplos Depósitos (*Multi-Depot Multiple Traveling Salesman Problem* – MDMTSP) e foi estudado por Chan e Baker (2005) e Benavent e Martínez (2011).

Existem ainda variações de outras naturezas, tais como limitação no número de vértices (ou na soma dos comprimentos das arestas) de cada ciclo; restrições de janela de tempo; custos diferentes para usar cada vértice como depósito; entre outras.

Suponha que, além de múltiplos ciclos disjuntos, queremos encontrar ainda um outro ciclo que interligue-os, isto é, cada vértice do ciclo interligador pertence também a exatamente um ciclo regular. Para complicar, não é dado na entrada o conjunto de depósitos (vértices que pertencem ao ciclo interligador); descobri-lo faz parte da busca pela solução ótima.

A fim de tornar o problema mais adequado a aplicações práticas, decidimos introduzir uma restrição quanto ao número máximo de vértices de cada ciclo regular. Essa cardinalidade máxima é uma constante inteira C, dada na entrada. Esse aspecto é análogo ao de um *VRP* de capacidade de veículos C e clientes com demanda unitária. No exemplo que usamos de transporte urbano, isso corresponde a proibir que um ônibus faça inúmeras paradas ao longo de seu trajeto, o que prejudicaria sua mobilidade. Essa restrição tem por consequência positiva evitar que a solução ótima seja composta por vários ciclos regulares pequenos, um único ciclo regular grande (contendo todos os demais vértices) e o ciclo interligador. Nesse caso, o problema torna-se muito parecido com o *TSP*, precisando (grosso modo) apenas decidir os ciclos pequenos e depois resolver o *TSP* para o ciclo regular grande, o que não seria interessante.

Além disso, as arestas do ciclo interligador são de natureza diferente das outras arestas, representando, em geral, vias mais rápidas. Portanto, foi introduzido um parâmetro α para multiplicar o custo de uma aresta na solução se ela for interligadora. O valor α é um fator racional constante, dado na entrada. Em princípio, pode parecer natural escolher valores de

$\alpha > 1$, pois a aresta interligadora seria mais cara para ser construída, por exemplo, no caso de meios de transporte expressos, como trem. No entanto, como toda viagem entre dois bairros envolveria o uso do transporte expresso, este seria muito mais utilizado, tendo um fator de ocupação muito maior e, por isso, a longo prazo é mais barato. Por isso, foram considerados valores de α entre 0 e 1. Esse raciocínio está alinhado com o critério utilizado em outros artigos da área.

Dessa forma, temos o Problema integrado de Localização e Roteamento com transporte entre concentradores e relação de Muitos-para-Muitos – LRCMM (*Many-to-many location-routing with inter-hub transport*). A entrada do problema consiste de um grafo métrico G de n vértices, uma função c_e de custo nas arestas, e três constantes: a capacidade C, o fator α e o número de concentradores k. As Figuras 2.1 e 2.2 são exemplos de soluções viáveis para o LRCMM, com $n = 29$, $k = 8$, $C = 5$ e $\alpha = 1$, para as duas versões analisadas: respectivamente, sem e com ciclos degenerados (ciclos com menos de três vértices).

Muitos problemas relacionados, derivados do Problema de Roteamento de Veículos, predeterminam, na entrada do problema, um ou mais vértices que serão depósitos. No nosso caso, essa determinação faz parte do problema, o que torna-o muito mais difícil. Em algumas instâncias que demoram horas para serem resolvidas, se tomarmos da solução final quais os depósitos e passarmos essa informação para o resolvedor, a execução do programa passa a ser feita em poucos minutos, o que mostra a importância dessa decisão. Além disso, pelo fato de termos depósitos não pré-determinados, muitos cortes conhecidos do Problema de Roteamento de Veículos deixam de ser aplicáveis ou, quando o são, precisam ser adaptados, às vezes perdendo bastante força, como mostrado no Capítulo 3.

Este nosso estudo resultou na publicação de um artigo no congresso LISS 2014 – *International Conference on Logistics, Informatics and Services Sciences*. O artigo apresenta o problema e alguns dos resultados computacionais obtidos. Além disso, estamos elaborando também um outro artigo a ser enviado para uma revista científica, com comparações de desempenho de algumas heurísticas para esse problema.

Recentemente, foi publicado um artigo por de Camargo *et al.* (2013) estudando um problema semelhante, em termos de ser um problema de localização e roteamento com circuito interligando os depósitos. É mais geral que o nosso problema em alguns aspectos: permite ciclos com um único cliente; aceita múltiplos ciclos em cada depósito; e recebe na entrada a demanda de cada cliente e o custo de abrir um depósito em um dado vértice. Por outro lado, tem um ponto que facilita substancialmente: a entrada já define o papel de cada vértice (cliente ou depósito). São dados na entrada um conjunto de clientes e um conjunto de candidatos a depósitos. Isto é, um programa que resolva esse problema deve decidir quais candidatos serão de fato depósitos, mas nenhum dos candidatos (escolhido ou não) terá qual-

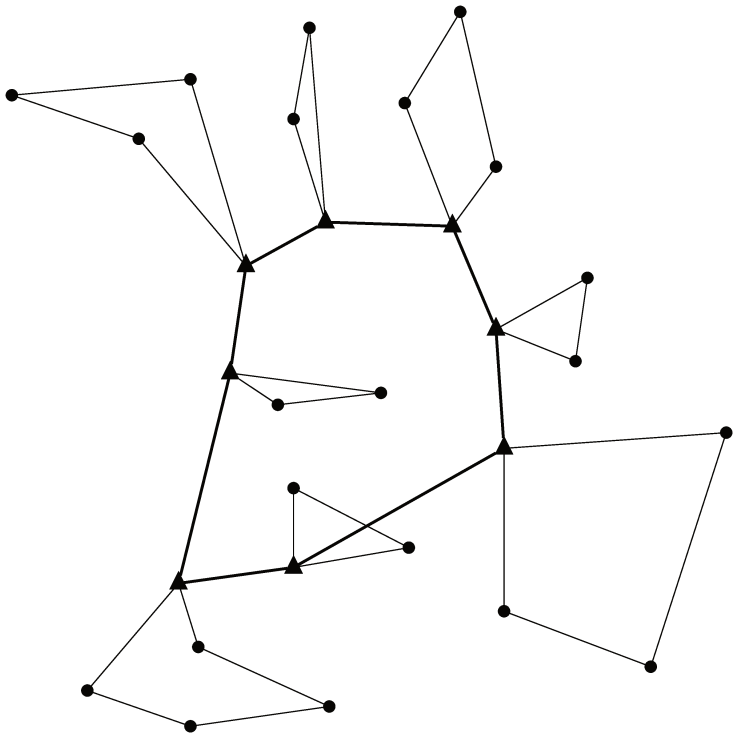


Figura 2.1: Exemplo de solução sem ciclos degenerados.

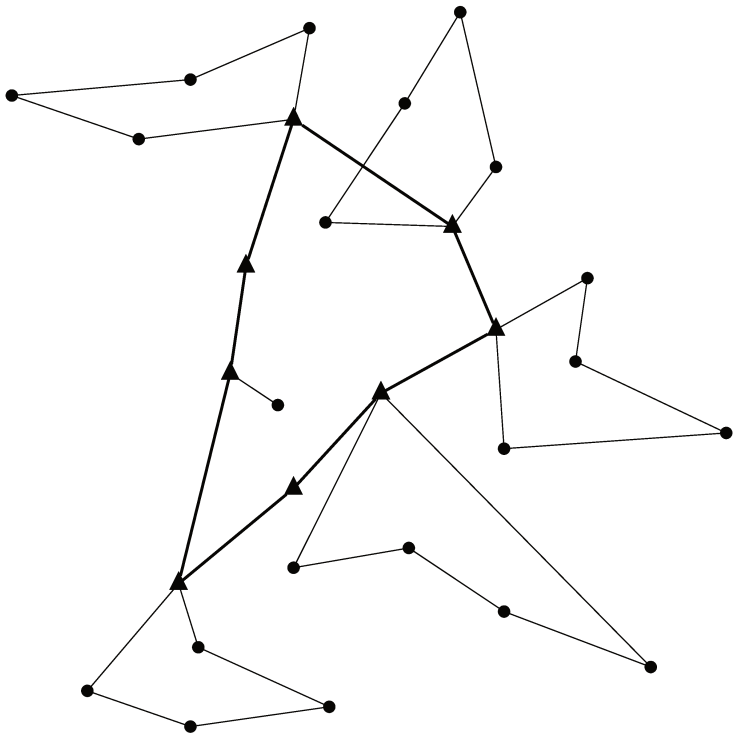


Figura 2.2: Exemplo de solução com ciclos degenerados.

quer demanda e, além disso, todos os vértices do conjunto de clientes serão necessariamente clientes. Esses fatos possibilitam o uso de cortes mais variados e mais fortes, como mostrado mais adiante.

Ainda mais recentemente, foi publicado por Rodríguez-Martín *et al.* (2014) um artigo de localização e roteamento de vértices com concentradores, porém com uma função-objetivo mais sofisticada, incorporando custos das rotas, custos de abertura de concentradores e custos do fluxo enviado entre pares de vértices. O programa por eles implementado foi capaz de resolver a maioria das instâncias de até 50 vértices dentro do tempo limite de duas horas.

Na perspectiva de problemas de localização e roteamento, como analisados por Lopes *et al.* (2013), temos um problema de localização e roteamento com relação de muitos-para-muitos com um subconjunto de K vértices (os concentradores) organizados em um circuito interno tal que, para cada concentrador, há um ciclo local contendo clientes que são servidos por ele. O objetivo é minimizar o custo total de todas as rotas, localizar os concentradores e atribuir clientes a concentradores com as seguintes restrições: todos os concentradores precisam estar conectados, isto é, há uma aresta entre todo par de concentradores; coleta e entrega podem não acontecer ao mesmo tempo, então um cliente pode ser visitado mais de uma vez; veículos têm capacidade e rotas são restringidas por um tempo limite de viagem; depósitos podem ter mais de um ciclo; e não há fator de desconto α.

Portanto, observando a literatura, nós abordamos uma variante do Problema de Localização e Roteamento com concentradores e relação muitos-para-muitos – LRCMM (*Many-to-many hub location-routing problem*) que foi proposto por Nagy e Salhi (1998) e possui aplicações na indústria de carga, incluindo transporte de pacotes, passageiros, aplicações postais, e indústria de bebidas – veja Kuby e Gray (1993) e Lin *et al.* (2012) – e em sistemas de telecomunicação, como em Wang *et al.* (2006). Note que o LRCMM é $\mathcal{NP}$-difícil já que possui dois subproblemas $\mathcal{NP}$-difíceis: Problema de Localização de Instalações e Problema do Caixeiro Viajante, como mostrado por Garey e Johnson (1979).

O artigo de Nagy e Salhi (1998) abordou o problema com uma heurística que resolve um problema de localização usando um método de adicionar/remover/deslocar, aprimorado com uma rotina de Busca Tabu, mas observando possíveis rotas desses concentradores resolvendo um problema de roteamento de múltiplos veículos de maneira heurística, realizando uma inter-relação precisa entre localização e roteamento. Também foram apresentadas uma formulação inteira e uma discussão de como ela pode resolver outros problemas relacionados, mas nenhum resultado experimental foi apresentado. Ao mesmo tempo, somente um exemplo ilustrativo foi resolvido com a heurística e os próprios autores comentam sua lentidão.

Observando a recente taxonomia de problemas de localização e roteamento de acordo com Lopes *et al.* (2013), apenas quatro trabalhos foram publicados sobre o LRCMM, incluindo o

estudo preliminar feito por Nagy e Salhi (1998). Bruns *et al.* (2000) consideraram o LRCMM, porém sem resolver explicitamente o problema de roteamento. Ao invés disso, propuseram uma aproximação que reduz o problema original para um simples problema de localização e que busca localizar os concentradores e alocar-lhes clientes. Esta estratégia foi usada pelos autores a fim de fornecer soluções para os correios da Suíça sob diferentes cenários. Wasner e Zaäpfel (2004) consideraram um estudo de caso real dos correios da Áustria modelado por uma formulação não-linear, mas resolvido com uma abordagem paralela com diferentes rotinas de busca local combinadas com laços de *feedback* para diminuir o custo da solução.

A estratégia primeiro determina o número e localização dos concentradores e atribui áreas postais a cada um deles, de modo que os custos dos concentradores podem ser obtidos. Depois, rotas de coleta e entrega são determinadas para os concentradores, e então o custo da solução é calculado. Laços de *feedback* servem para diminuir o custo da solução reatribuindo áreas postais para os concentradores, combinando rotas, e alterando o número e localização dos concentradores. Além disso, na variante deles do LRCMM, clientes podem atender diretamente outros clientes, ou seja, pode haver uma aresta entre um par de clientes. Ademais, é considerado um concentrador central pelo qual passa todo o tráfego entre concentradores, e coleta e entrega ocorrem no mesmo instante.

Embora o trabalho de Karaoglan *et al.* (2012) pareça relacionado a problemas de localização e roteamento com relação de muitos-para-muitos na classificação de Lopes *et al.* (2013), tal trabalho considerou o problema de localização e roteamento com restrições de coleta e entrega simultâneas nas quais não são permitidas rotas entre concentradores. Em último caso, pode ser considerado um caso especial do LRCMM no qual não há rotas entre concentradores, embora sem uma contribuição específica a este.

No artigo de Çetiner *et al.* (2010), o LRCMM é resolvido com um procedimento de duas fases. As localizações dos concentradores são determinadas bem como as alocações de clientes a concentradores e, então, rotas são calculadas. O procedimento alterna entre as fases atualizando os concentradores e as rotas a fim de reduzir os custos totais. Experimentos com o correio turco e instâncias da literatura mostraram a competitividade desse algoritmo. Nessa variante do problema, a coleta e a entrega ocorrem juntas quando um cliente é visitado, e clientes podem ser atendidos por mais de um concentrador.

Recentemente, de Camargo *et al.* (2013) propuseram uma extensa formulação inteira para o LRCMM que combina modelos do problema de localização de concentradores de alocação única e do problema do caixeiro viajante. Sua formulação usa variáveis inteiras de quatro índices combinadas com variáveis contínuas de cinco índices, mas uma vez que as variáveis são fixadas, surgem dois subproblemas simples. Esta característica permite que seja aplicada a decomposição de Benders, de modo que instâncias com até 100 vértices sejam

resolvidas, embora tenham sido reportados resultados para apenas uma instância com 100 vértices e exigiu aproximadamente 46 horas de processamento. Além disso, os resultados são concentrados em instâncias com até 50 vértices e para diferentes valores de α, limitado a duas horas de tempo de computação. Relataram que, à medida que α aumenta, o número de concentradores diminui e o tempo para resolver as instâncias aumenta.

Para outras variantes de problemas de localização e roteamento, o leitor pode seguir a classificação recente apresentada em Lopes *et al.* (2013), além das pesquisas feitas por Min *et al.* (1998) e por Nagy e Salhi (2007) apresentando novas tendências para futuros trabalhos. Heurísticas para o problema de localização e roteamento foram desenvolvidas por Yu *et al.* (2010), Nguyen *et al.* (2012), Mehrjerdi e Nadizadeh (2013) e Zarandi *et al.* (2013), respectivamente, para os casos com capacidade, com dois níveis, com demandas *fuzzy* e com janelas de tempo combinadas com parâmetros *fuzzy*.

Uma outra variante foi introduzida por Andrade *et al.* (2013) chamada Problema do Caixeiro Viajante Múltiplo com k depósitos interligados, que visa problemas de telecomunicação, com o desenvolvimento de um Algoritmo Genético de Chaves Aleatórias Viciadas (*Biased Random-key Genetic Algorithms* – BRKGA) usando rotinas de busca local que combinam operações de troca e otimização de subciclos. Outros estudos, como os de Belenguer *et al.* (2011), Contardo *et al.* (2012) e Contardo *et al.* (2013), consideram abordagens exatas que combinam *branch-and-bound* com algoritmos de planos de corte, tal qual nós apresentamos a seguir, mas para a variante capacitada do problema de localização e roteamento.

Capítulo 3

Localização e Interligação de Concentradores com Roteamento de Veículos

3.1 Introdução

O Problema de Localização de Concentradores com Roteamento de Veículos (LRCMM) é uma generalização do CVRP em que existem k depósitos e eles formam um ciclo adicional.

Dados um grafo de n vértices e um inteiro k, o objetivo é formar k ciclos de vértices, além de um ciclo adicional (interligador) formado por um vértice de cada um deles, de modo a minimizar a soma das arestas escolhidas. Há também os parâmetros de entrada α, que é um fator nos custos das arestas do ciclo interligador, e C, que representa o número máximo de vértices de cada ciclo local (não se aplica ao ciclo interligador, pois este possui exatamente k vértices).

Trata-se de um problema $\mathcal{NP}$-difícil. Uma maneira de se provar isso, supondo a versão que permite ciclos com menos de três vértices, é fazendo a seguinte redução a partir do TSP: dada uma instância do TSP com n vértices, use-a como entrada do nosso problema junto com os parâmetros $k = n$, $\alpha = 1$ e $C = 1$. Nessa solução, devido ao limite do número de vértices (capacidade), cada ciclo regular consistirá de apenas um concentrador sem clientes. O ciclo interligador, com $k = n$ vértices, será necessariamente a solução do TSP original. Desse modo, se houver um algoritmo de tempo polinomial que resolva o LRCMM, temos um algoritmo de tempo polinomial para o TSP. Logo, o LRCMM é $\mathcal{NP}$-difícil.

3.2 Notação e nomenclatura

Seja $G = (V, E)$ um grafo simples, completo, não-orientado, com conjunto V de vértices ($|V| = n$) e E de arestas. Estas últimas correspondem a pares de vértices, possivelmente ordenados. Neste trabalho, tratamos apenas de arestas não-orientadas. Apesar disso, é mantida a notação (u, v) para as arestas. Com isso, (u, v) e (v, u) referem-se à mesma aresta.

Define-se uma função $c : E \to \mathbb{Q}^+$, tal que c_e representa o custo da aresta e. Dado subgrafo G' de G, denotamos por $V(G')$ e $E(G')$ seu conjunto de vértices e de arestas, respectivamente.

Um ciclo $\mathcal{C}$ é um subgrafo de G cujos vértices $V(\mathcal{C})$ podem ser ordenados como $v_1, \ldots, v_t$ de forma que $\{(v_i, v_{(i \bmod t)+1}), 1 \leq i \leq t\} = E(\mathcal{C})$. Um ciclo é dito *degenerado* caso possua menos de três vértices.

Analogamente, um caminho P é um subgrafo de G cujos vértices $V(P)$ podem ser ordenados como $v_1, \ldots, v_t$ de modo que $\{(v_i, v_{i+1}), 1 \leq i \leq t\} = E(P)$. Portanto, o subgrafo $(V(P), E(P) \cup (v_1, v_t))$ é um ciclo. Os vértices v_1 e v_t são ditos *extremidades* de P, enquanto que os demais são ditos *vértices internos* a P.

O conjunto de todos os caminhos que ligam o vértice i ao vértice j será denotado por $\mathcal{P}_{ij}$, enquanto o conjunto de todos os caminhos no grafo será denotado por $\mathcal{P}$, considerando que o contexto será claro a respeito de qual o grafo ao qual os caminhos pertencem.

Um grafo G é *completo* se todo par de vértices distintos está ligado por uma aresta, isto é, $(i, j) \in E(G), \forall i, j \in V, i \neq j$. Diz-se que um grafo completo G *satisfaz a desigualdade triangular* (ou que é *métrico*) quando $c_{(u,v)} + c_{(v,w)} \geq c_{(u,w)}, \forall u, v, w \in V(G)$.

Conjuntos $S_1, ..., S_r$ são ditos *disjuntos dois a dois* (ou *mutuamente disjuntos*) se $S_t \cap S_{t'} = \emptyset, \forall t, t' \in \{1, 2, ..., r\}, t \neq t'$. Consequentemente, dizer que dois subgrafos são *vértice-disjuntos* significa que eles não possuem vértices em comum. O termo *aresta-disjuntos* é análogo para arestas.

Uma partição de um conjunto S é uma coleção $S_1, ..., S_r$ de subconjuntos de S, não-vazios, mutuamente disjuntos, e cuja união corresponde a todos os elementos de S, isto é, $S_1 \cup ... \cup S_r = S$.

Para um subconjunto de vértices $S \subset V$, denotamos por $\overline{S}$ seu complemento em relação a V, ou seja, $\overline{S} = V - S$, e dizemos que o par $\left(S, \overline{S}\right)$ é um *corte* de G. O conjunto de arestas da fronteira de S (arestas com exatamente uma extremidade em S) será denotado por $\delta(S)$. Formalmente, $\delta(S) = \{(i, j) \in E \mid i \in S, j \notin S\}$. Quando o conjunto S é unitário, simplificamos $\delta(\{v\})$ para $\delta(v)$.

Especificamente para o LRCMM, o ciclo que intercepta todos os outros ciclos será chamado de *ciclo interligador*, enquanto os outros serão chamados de *ciclos regulares*. Por extensão, os adjetivos *interligador* e *regular* também serão usados para vértices e arestas que

pertençam a ciclos dos respectivos tipos. Os vértices interligadores poderão ser chamados de *depósitos* ou de *concentradores (hubs)*, de acordo com a nomenclatura utilizada na literatura em problemas similares.

Um subgrafo de G é *viável* se possui as propriedades desejadas das soluções do problema, isto é, se suas arestas formam exatamente k ciclos mutuamente disjuntos, passando por todos os vértices de G, cada um com no máximo C vértices; e um outro ciclo, este contendo exatamente um vértice de cada um dos outros ciclos.

Em uma formulação de Programação Linear, chamamos de *valoração* uma atribuição específica de valores para cada variável da formulação. Uma *valoração válida* é aquela que satisfaz todas as restrições do problema, exceto possivelmente a otimalidade da função objetivo. Sendo assim, um problema de minimização consiste em encontrar uma valoração válida cujo valor da função objetivo é mínimo, isto é, uma *solução ótima* do problema.

Ao longo do texto, usamos os termos *inequação* e *desigualdade* como sinônimos, indicando sempre uma relação de maior-ou-igual ($\geq$) ou de menor-ou-igual ($\leq$).

3.3 Definição do problema

A entrada do LRCMM possui a seguinte forma: são dados um grafo $G = (V, E)$, uma função $c : E \to Q^+$ de custo das arestas de G, um racional α, e inteiros k e C.

Uma solução viável geral consiste de $k + 1$ ciclos contidos em G: um ciclo *interligador* I, dado por *depósitos* $t_1, t_2, ..., t_k$, e k ciclos *regulares* $\mathcal{C}_1, ..., \mathcal{C}_k$. Para simplificar a notação, escrevemos $\mathcal{C}_i$ em vez de $\mathcal{C}_{t_i}$. Cada depósito $t_i \in I$ pertence a $\mathcal{C}_i$ e a nenhum outro ciclo regular. Os ciclos regulares $\mathcal{C}_1, ..., \mathcal{C}_k$ formam uma partição de V. Cada ciclo regular tem cardinalidade no máximo C.

O custo $\mathcal{F}(x)$ de uma solução viável x é dado pela soma dos custos c_e das arestas de seus ciclos, com fator α para as arestas do ciclo interligador. O problema consiste em encontrar uma solução viável de custo mínimo.

Para este estudo, foi considerada a versão do problema em grafos métricos, com ciclos regulares de capacidade máxima dada por um parâmetro de entrada C. Consideramos principalmente a versão em que todos os ciclos devem ser não-degenerados, ou seja, devem ter pelo menos três vértices (um concentrador e dois clientes). A formulação para a versão com ciclos degenerados também é apresentada, bem como resultados experimentais.

Problemas similares, como em Lysgaard *et al.* (2004), permitem que clientes possuam diferentes demandas. No LRCMM, fizemos uma simplificação ao considerar que todos os clientes têm a mesma importância (equivalente a usar demanda unitária para os clientes). Isso nos permite usar, sem perdas significativas na dificuldade do problema, entradas padrão

da TSPLIB, conjunto descrito por Reinelt (1991) de instâncias para o Problema do Caixeiro Viajante.

3.4 Formulação em Programação Linear Inteira

Esta formulação, para a versão do problema sem ciclos degenerados, possui três conjuntos de variáveis, todas binárias. O primeiro conjunto, dado pelas variáveis x_e, para cada aresta $e \in E$, define se uma aresta é interligadora:

$$x_e = \begin{cases} 1 & \textit{se a aresta } \mathbf{e} \textit{ é interligadora}, \\ 0 & \textit{caso contrário}. \end{cases}$$

O segundo conjunto, dado pelas variáveis y_v, para cada vértice $v \in V$, indica se um vértice é concentrador:

$$y_v = \begin{cases} 1 & \textit{se o vértice } \mathbf{e} \textit{ é concentrador}, \\ 0 & \textit{caso contrário}. \end{cases}$$

Por fim, o terceiro conjunto, formado pelas variáveis z_e, para cada aresta $e \in E$, define se uma aresta é regular:

$$z_e = \begin{cases} 1 & \textit{se a aresta } \mathbf{e} \textit{ é regular}, \\ 0 & \textit{caso contrário}. \end{cases}$$

Dadas essas variáveis, apresentamos a seguir uma formulação para o problema e, em seguida, explicamos cada restrição:

$$\text{minimizar} \sum_{e \in E} c_e \left(\alpha x_e + z_e\right) \tag{3.1}$$

sujeito a:

$$\sum_{v \in V} y_v = k \tag{3.2}$$

$$\sum_{e \in \delta(v)} z_e = 2, \ \forall v \in V \tag{3.3}$$

$$\sum_{e \in \delta(v)} x_e = 2y_v, \ \forall v \in V \tag{3.4}$$

$$x_e + z_e \leq 1,\ \forall e \in E \tag{3.5}$$

$$\sum_{e \in \delta(S)} (x_e + z_e) \geq 2,\ S \subset V,\ |S| \geq 1 \tag{3.6}$$

$$\sum_{e \in \delta(S)} x_e \geq 2\left(y_i + y_j - 1\right),\ \forall i, j \in V,\ S \subset V,\ |S \cap \{i, j\}\,| = 1 \tag{3.7}$$

$$\sum_{e \in E(P)} z_e + \sum_{v \in V(P)} y_v \leq |V\left(P\right)|,\ \forall i, j \in V,\ i \neq j,\ P \in \mathcal{P}_{ij} \tag{3.8}$$

$$\sum_{e \in \delta(S)} z_e \geq 2\frac{\sum\limits_{i \in S}(1 - y_i)}{C - 1} - 2\sum_{i \in S} y_i,\ \forall S \subset V,\ |S| \geq 2 \tag{3.9}$$

$$x_e \in \{0, 1\},\ \forall e \in E \tag{3.10}$$

$$y_i \in \{0, 1\},\ \forall i \in V \tag{3.11}$$

$$z_e \in \{0, 1\},\ \forall e \in E. \tag{3.12}$$

A restrição (3.2) fixa a quantidade de depósitos na solução. As restrições (3.3) são as restrições de grau para todos os vértices (como as do *TSP*) e fazem com que cada vértice tenha exatamente duas arestas regulares incidentes. Analogamente para as restrições (3.4) em relação às arestas interligadoras, quando o vértice é um depósito; caso contrário, não são permitidas tais arestas. As restrições (3.5) indicam que uma mesma aresta não pode ser regular e interligadora.

Para a eliminação de subciclos, também temos duas classes de desigualdades: as do tipo (3.6) asseguram que o grafo todo seja conexo, ao garantir a existência de pelo menos duas arestas em qualquer corte, enquanto que as do tipo (3.7) atuam de modo similar, porém apenas nos depósitos e nas arestas interligadoras. Quando pelo menos um dos vértices é regular, a desigualdade é trivialmente satisfeita.

As duas classes restantes de restrições são a de desigualdades de *concentrador único por ciclo* (3.8) e a de *desigualdades de capacidade* (3.9). Como estes dois tipos são consideravelmente mais sofisticados que os outros, estudamos cada um deles em uma sessão dedicada.

Nesta formulação existem duas variáveis (uma x e uma z) para cada aresta, além de uma

variável y para cada vértice. Portanto, o número total de variáveis é $2|E|+|V| = n(n-1)+n = n^2$. O número de restrições inseridas antes do começo da execução do algoritmo branch-and-cut é de $|E|$ para as restrições (3.5); $|V|$ para cada uma das classes (3.3) e (3.4); e 1 do tipo (3.2), totalizando $|E| + 2|V| + 1 = \frac{n(n-1)}{2} + 2n + 1 = \frac{n^2+3n+2}{2}$. Todas as outras classes de restrições são inseridas à medida que se tornam necessárias.

Note que, pelas restrições (3.4), podemos substituir y_v por $\frac{1}{2}\sum_{e\in\delta(v)} x_e$ em todas as outras restrições. Isto eliminaria todas as variáveis y, o que poderia agilizar um pouco a resolução do programa linear. Entretanto, preferimos aqui manter essas variáveis para facilitar o entendimento por parte do leitor.

3.4.1 Desigualdades de concentrador único por ciclo

Esta classe de desigualdades tem por finalidade evitar que um ciclo regular passe por dois depósitos, sejam eles interligados ou não. A situação que queremos evitar é ilustrada na Figura 3.1.

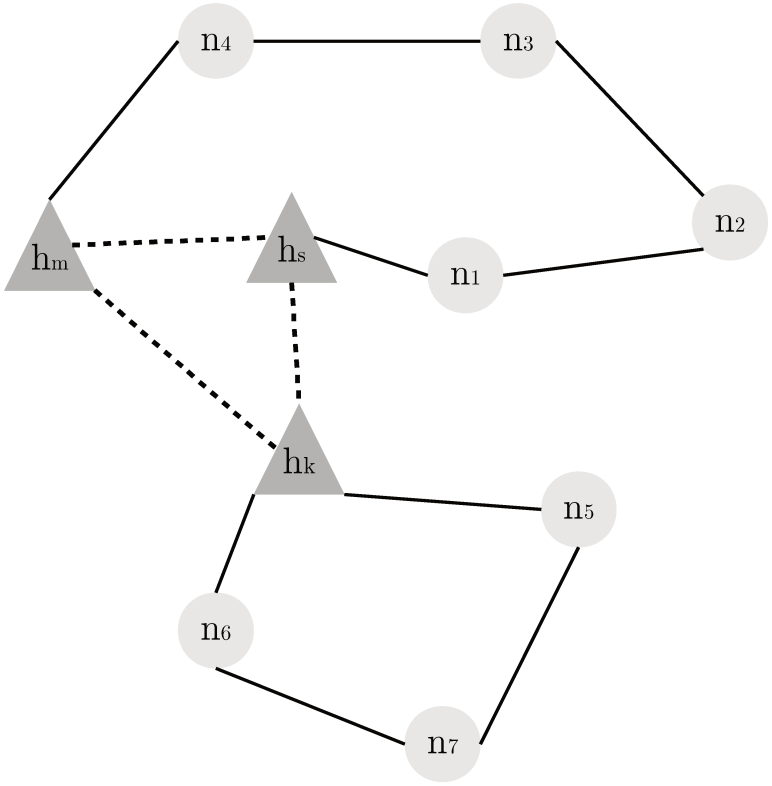


Figura 3.1: Exemplo de configuração eliminada por uma restrição (3.8).

A ideia da desigualdade é eliminar todo caminho cujas extremidades sejam depósitos e cujas arestas sejam todas regulares. Aqui demonstramos que elas de fato funcionam.

Proposição 3.1. *As desigualdades* (3.8) *eliminam todos os casos em que a intersecção entre o conjunto de vértices do ciclo interligador e o de um outro ciclo tenha pelo menos dois elementos.*

Demonstração. Suponha que um ciclo interligador possua dois depósitos em um mesmo ciclo regular. Dentre os depósitos desse ciclo regular, sejam i e j dois deles consecutivos no ciclo regular. Logo, eles estão ligados por um caminho $P \in \mathcal{P}_{ij}$ formado totalmente por arestas regulares, sendo que suas duas extremidades (i e j) são depósitos e todos os seus vértices internos são regulares. Portanto:

$$\sum_{e \in E(P)} z_e + \sum_{v \in V(P)} y_v = |E(P)| + 2 = |V(P)| + 1 > |V(P)|,$$

o que não é possível de acordo com as desigualdades (3.8). □

Proposição 3.2. *Se um subgrafo de G não satisfaz todas as restrições* (3.8)*, então existe alguma restrição* (3.8)*, não satisfeita por ele, da seguinte forma: um caminho P com depósitos em suas extremidades; todos os vértices internos regulares; e todas as arestas regulares.*

Demonstração. Dado um subgrafo de G, considere todas as restrições (3.8) não satisfeitas por ele, ou seja, tais que

$$\sum_{e \in E(P)} z_e + \sum_{v \in V(P)} y_v \geq |V(P)| + 1. \tag{3.13}$$

para algum caminho $P \in \mathcal{P}$. Dessas desigualdades, tome aquela com caminho P mais curto (em caso de empate, escolha qualquer uma). Sejam i e j as extremidades de P, ou seja, $P \in \mathcal{P}_{ij}$. Nessa desigualdade, o vértice i deve ser um depósito e seu vizinho i' em P ser regular; caso contrário, poderíamos remover i e obter um novo caminho, menor que o mínimo, que violasse a desigualdade, pois do lado esquerdo subtrairíamos $z_{ii'} + y_i \leq 1$, e do lado direito subtrairíamos 1, mantendo o sinal a desigualdade. O mesmo raciocínio vale para o vértice j.

Além disso, todos os vértices internos de P são regulares. Suponha por absurdo que um vértice interno h de P seja depósito. Como o caminho P é mínimo, os subcaminhos de P ligando i a h e h a j, denotados aqui por P_{ih} e P_{hj} devem satisfazer suas respectivas desigualdades (3.8). Logo, as duas inequações a seguir são válidas:

$$\sum_{e \in E(P_{ih})} z_e + \sum_{v \in V(P_{ih})} y_v \leq |V(P_{ih})|$$

e

$$\sum_{e \in E(P_{hj})} z_e + \sum_{v \in V(P_{hj})} y_v \leq |V\left(P_{hj}\right)|.$$

Somando-as, temos:

$$\sum_{e \in E(P)} z_e + \sum_{v \in V(P)} y_v + y_h \leq |V\left(P\right)| + 1$$

Absurdo, pois contradiz (3.13). □

Proposição 3.3. *As desigualdades* (3.8) *não eliminam subgrafos de* G *que queremos que sejam viáveis.*

Demonstração. Se um subgrafo de G é eliminado por alguma restrição (3.8), então, pela Proposição (3.2), ele possui um caminho de arestas regulares ligando dois depósitos. Como nos subgrafos viáveis os depósitos são desconexos exceto pelo ciclo interligador, então de fato o subgrafo em questão é inviável e corretamente descartado. □

3.4.2 Desigualdades de capacidade

Esta classe de desigualdades é adaptada de Lysgaard *et al.* (2004). A inequação naquele artigo, convertendo-se a notação e considerando demanda unitária, possui a forma

$$\sum_{e \in \delta(S)} z_e \geq 2 \left\lceil \frac{|S|}{C} \right\rceil, \ \forall S \subset V, \ |S| \geq 2 \tag{3.14}$$

e significa que, em qualquer conjunto de clientes, o número de ciclos passando por seu interior deve ser suficiente para que todos sejam visitados, respeitando a capacidade dos ciclos. Funciona da seguinte maneira: se cada ciclo possui capacidade C e há $|S|$ clientes, então há no mínimo $\left\lceil \frac{|S|}{C} \right\rceil$ ciclos, cada um com pelo menos duas arestas no corte. Portanto, o total de arestas no corte é pelo menos $2 \left\lceil \frac{|S|}{C} \right\rceil$, o que origina a desigualdade (3.14).

No caso do LRCMM, há algumas dificuldades adicionais. Não se sabe de antemão quais vértices são depósitos. Portanto, o número de clientes em S passa a ser $\sum_{i \in S} (1 - y_i)$ em vez de $|S|$. Isso já impossibilita o arredondamento para cima ($\lceil \ \rceil$), uma vez que ela iria conter variáveis e tal operação não é permitida em Programação Linear. O que a equação (3.14) considera capacidade de um veículo (o número de clientes que ele pode atender), nós consideramos a capacidade (cardinalidade máxima) de um ciclo; logo, trocamos C por $C - 1$.

Por fim, note que a ausência de definição antecipada da finalidade (cliente ou depósito) de cada vértice traz mais uma complicação: para cada depósito que o conjunto S vier a conter, um depósito externo a menos será necessário para atender aos clientes em S, o que

corresponde a duas arestas a menos na fronteira de S. Por isso, subtraímos $2\sum_{i \in S} y_i$ do membro direito da inequação e, desse modo, foi obtida a inequação (3.9). Infelizmente, essa inequação é bem menos apertada que a inequação (3.14), pois as duas modificações (remoção do arredondamento e subtração de termos não-negativos no lado menor da desigualdade) pioram sua eficácia.

3.4.3 Corretude da formulação

Nesta subseção, argumentamos a fim de mostrar que a formulação apresentada define corretamente o problema, ou seja, que ela de fato aceita todos os subgrafos de G que queremos que sejam viáveis e rejeita todos os outros. Queremos que cada valoração viável corresponda a exatamente um subgrafo de G viável, e vice-versa.

Primeiro, considere uma valoração viável qualquer. É fácil obter o subgrafo correspondente, basta tomar todas as arestas e tais que $x_e + z_e = 1$. Ele é viável, de acordo com a definição, pois: possui exatamente k depósitos, por (3.2); todos eles pertencem a um único ciclo (interligador), de acordo com (3.4) e (3.7), e (3.5) garante que nenhuma aresta seja regular e interligadora. Já (3.3) faz com que o conjunto V esteja particionado em ciclos, e garante que cada depósito pertença a exatamente um ciclo. Além disso, como essa restrição exige duas arestas regulares em cada depósito, é impossível que haja ciclos degenerados, pois estes ocorreriam em depósitos com apenas uma ou nenhuma aresta regular.

Por (3.6), a solução é conexa. Como os ciclos regulares são disjuntos, todos eles estão ligados pelo ciclo interligador, que, tendo k vértices, consegue ligar no máximo k ciclos. Por (3.8), existe no máximo um depósito por ciclo; logo, o número de ciclos é no mínimo o número de depósitos, k. Portanto, o número de ciclos regulares é necessariamente igual a k, cada um com exatamente um depósito. Cada ciclo regular possui no máximo C vértices, de acordo com (3.9).

Considere agora um subgrafo de G viável qualquer. Para obter a valoração que a ele corresponde, procedemos da seguinte maneira: para todos os vértices i que possuem mais de duas arestas incidentes, $y_i = 1$; para todas as arestas $e = (i, j)$ que liguem vértices com $y_i = y_j = 1$, temos $x_e = 1$; para todas as demais arestas, $z_e = 1$; por fim, todas as variáveis restantes têm valor zero. Note que há uma única valoração que pode ser obtida.

Essa valoração é válida. A desigualdade (3.5) é trivialmente satisfeita, pois só foi fixado $z_e = 1$ nas arestas em que não ocorre $x_e = 1$ e que, portanto, possuem $x_e = 0$. A desigualdade (3.3) é satisfeita porque fixamos $z_e = 1$ para todas as arestas dos ciclos disjuntos (e só para elas) e todo vértice pertence a um deles. Similarmente, a desigualdade (3.4) é satisfeita pois, por construção, todo vértice i com $y_i = 1$ pertence ao ciclo interligador e, portanto, possui

exatamente duas arestas (digamos, (i, i') e $(i, i")$) com valor $x_{ii'} = x_{ii"} = 1$, enquanto que toda aresta incidente a qualquer vértice i com $y_i = 0$ possui $x_e = 0$.

As restrições 3.7 são facilmente satisfeitas se não ocorrer $y_i = y_j = 1$, pois o lado direito da inequação seria não-positivo. No outro caso, o conjunto S deve conter exatamente um entre os dois vértices considerados. Como ambos pertencem a um mesmo ciclo de arestas e com $x_e = 1$, há duas dessas arestas que cruzam a fronteira de S, satisfazendo a inequação.

Como o subgrafo viável é conexo e todo vértice tem pelo menos dois vizinhos (de acordo com (3.6)), então, qualquer que seja o conjunto S, haverá duas arestas cruzando sua fronteira, satisfazendo (3.6).

A equação (3.2) é imediata, já que o subgrafo possui k ciclos regulares, cada um com exatamente um vértice em comum com o ciclo interligador, e todos esses vértices (e apenas eles) possuem valor 1 em sua variável y_v correspondente.

A Proposição (3.3) já trata das inequações (3.8), enquanto que as do tipo (3.9) são explicadas na seção 3.4.2.

Por sua vez, as restrições (3.10), (3.11) e (3.12) são naturalmente satisfeitas.

3.4.4 Versão do problema com ciclos degenerados

Durante nosso estudo, foi considerada a possibilidade de aceitar ciclos degenerados nas soluções: ciclos com apenas um cliente, ou seja, uma única aresta regular ligada ao depósito, sendo utilizada para os dois sentidos da viagem que atende o cliente; e ciclos sem cliente, apenas com o depósito. A inclusão desses casos permite potencialmente obter soluções melhores. Do ponto de vista prático, uma solução dessas pode ou não ser aceitável, dependendo da aplicação. Por exemplo, é possível construir uma estação de metrô bem no centro da cidade, sem linhas de ônibus passando por ela, pois, sendo numa região suficientemente densa, beneficiaria muitas pessoas que morassem ou trabalhassem a uma pequena distância da estação, de modo que pudessem caminhar até seu destino. Por outro lado, para aplicações de rede, pode não fazer sentido implantar uma central de distribuição de dados em uma região que não possua usuários.

Uma solução viável para o caso com ciclos degenerados é dada na Figura 3.2. Veja que há depósitos isolados; depósitos com um único cliente; e depósitos com pelo menos dois clientes.

Para permitir ciclos degenerados, são necessárias algumas mudanças na formulação. As restrições 3.3 mudariam de

$$\sum_{e \in \delta(v)} z_e = 2, \ \forall v \in V$$

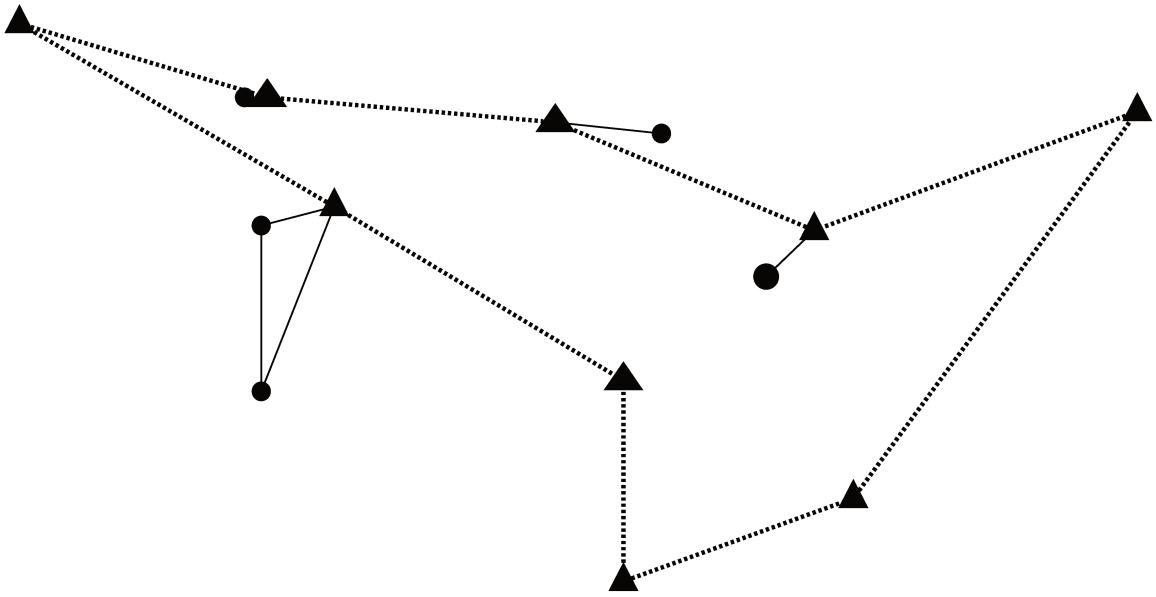


Figura 3.2: Solução viável apenas no problema que aceita ciclos degenerados.

para

$$\sum_{e \in \delta(v)} z_e \leq 2, \ \forall v \in V.$$

Isso permitiria muito mais possibilidades de valores fracionários nos nós da árvore de *Branch-and-cut*. Pior: mesmo considerando apenas valores inteiros, são criados outros casos permitidos. Poderia ocorrer de um vértice regular vir a não ter arestas (estar totalmente isolado), o que só seria evitado após a inserção das restrições (3.6), computacionalmente muito mais trabalhosas de encontrar, como explicado na seção 3.5.2. Além disso, poderia também haver um caminho de arestas regulares ligando dois vértices, cada um com exatamente uma aresta regular ($\sum_{e \in \delta(v)} z_e = 1$), sendo resolvido somente ao adicionarmos novas restrições do tipo (3.8), em caso de dois depósitos, ou (3.6), caso contrário. Ambas são também muito mais custosas de encontrar, como será mostrado nas seções 3.5.3 e 3.5.2, respectivamente.

Aqui apresentamos a versão completa da formulação que permite ciclos degenerados:

$$\text{minimizar} \sum_{e \in E} c_e \left(\alpha x_e + z_e + 2w_e\right) \tag{3.15}$$

sujeito a:

$$\sum_{v \in V} y_v = k \tag{3.16}$$

$$\sum_{e\in\delta(v)} (z_e + 2w_e) \leq 2,\ \forall v \in V \tag{3.17}$$

$$\sum_{e\in\delta(v)} x_e = 2y_v,\ \forall v \in V \tag{3.18}$$

$$x_e + z_e + w_e \leq 1,\ \forall e \in E \tag{3.19}$$

$$\sum_{e\in\delta(S)} (x_e + z_e + 2w_e) \geq 2,\ S \subset V,\ |S| \geq 1 \tag{3.20}$$

$$\sum_{e\in\delta(S)} x_e \geq 2\left(y_i + y_j - 1\right),\ \forall i,j \in V,\ S \subset V,\ |S \cap \{i,j\}\,| = 1 \tag{3.21}$$

$$\sum_{e\in E(P)} z_e + \sum_{v\in V(P)} y_v \leq |V\left(P\right)|,\ \forall i,j \in V,\ i \neq j,\ P \in \mathcal{P}_{ij} \tag{3.22}$$

$$\sum_{e\in\delta(S)} (z_e + 2w_e) \geq 2\frac{\sum\limits_{i\in S}(1 - y_i)}{C-1} - 2\sum_{i\in S} y_i,\ \forall S \subset V,\ |S| \geq 2 \tag{3.23}$$

$$w_e \leq y_i + y_j,\ \forall e = \{i,j\} \in E \tag{3.24}$$

$$z_e + w_e + y_i + y_j \leq 2,\ \forall e = \{i,j\} \in E \tag{3.25}$$

$$x_e \in \{0,1\},\ \forall e \in E \tag{3.26}$$

$$y_i \in \{0,1\},\ \forall i \in V \tag{3.27}$$

$$z_e \in \{0,1\},\ \forall e \in E \tag{3.28}$$

$$w_e \in \{0,1\},\ \forall e \in E. \tag{3.29}$$

Nota-se a inclusão de uma variável w_e para cada aresta $e \in E$, representando o uso da aresta E em um ciclo de cliente único. Dizemos que uma aresta com $w_e = 1$ é uma aresta *dupla*. A função objetivo passa a conter o termo $2c_e w_e$ para adicionar o custo de ida e de volta dessas arestas. As restrições inseridas ou alteradas em relação à formulação da seção anterior

são aquelas que aqui envolvem variáveis do tipo w. As restrições (3.3) foram alteradas de:

$$\sum_{e \in \delta(v)} z_e = 2, \ \forall v \in V$$

para (3.17):

$$\sum_{e \in \delta(v)} (z_e + 2w_e) \leq 2, \ \forall v \in V,$$

pois agora cada vértice pode possuir duas arestas comuns ou uma aresta dupla, ou mesmo nenhuma, já que agora admitimos concentradores sem clientes. As desigualdades (3.5), dadas por

$$x_e + z_e \leq 1, \ \forall e \in E,$$

transformam-se nas (3.19):

$$x_e + z_e + w_e \leq 1, \ \forall e \in E,$$

para manter a propriedade de cada aresta só poder ser de um único tipo. Como qualquer solução viável deve ser conexa considerando todos os tipos de arestas, a restrição (3.6):

$$\sum_{e \in \delta(S)} (x_e + z_e) \geq 2, \ S \subset V, |S| \geq 1$$

passa a ser (3.20):

$$\sum_{e \in \delta(S)} (x_e + z_e + 2w_e) \geq 2, \ S \subset V, |S| \geq 1.$$

Da mesma forma, o termo z_e da inequação (3.9):

$$\sum_{e \in \delta(S)} z_e \geq 2\frac{\sum_{i \in S}(1 - y_i)}{C - 1} - 2\sum_{i \in S} y_i, \ \forall S \subset V, \ |S| \geq 2$$

é trocado por $z_e + 2w_e$ em (3.23):

$$\sum_{e \in \delta(S)} (z_e + 2w_e) \geq 2\frac{\sum_{i \in S}(1 - y_i)}{C - 1} - 2\sum_{i \in S} y_i, \ \forall S \subset V, \ |S| \geq 2$$

porque as arestas que atendem os clientes podem ser regulares ou duplas.

As restrições que não possuem correspondentes da formulação anterior são de dois tipos: as (3.24) impedem que existam arestas duplas entre dois clientes, enquanto as do tipo (3.25) significam que dois concentradores só podem estar unidos por aresta interligadora.

Neste trabalho, focamos na versão sem ciclos degenerados. Entretanto, as explicações

sobre rotinas de separação e detalhes de implementação aplicam-se facilmente também à versão com ciclos degenerados, com pouca ou até mesmo nenhuma modificação.

3.5 Rotinas de Separação

Nesta seção, explicamos mais detalhes de cada restrição, incluindo os algoritmos de separação para aquelas que são adicionadas ao modelo durante a execução do programa. Denotamos aqui por $(\bar{x}, \bar{y}, \bar{z})$ a solução ótima da relaxação, e por $G^* = (V, E^*)$ o grafo suporte de G, com $E^* = \{e \in E \mid \bar{z}_e > 0\}$.

3.5.1 Restrições de Conectividade de Depósitos

As restrições do tipo (3.7) garantem que todo par de depósitos seja ligado por um caminho de arestas interligadoras. Em outras palavras, como todo depósito possui exatamente duas arestas interligadoras, as restrições mencionadas eliminam a possibilidade de essas arestas formarem subciclos. Como existe uma quantidade exponencial dessas restrições, elas não podem ser adicionadas todas no começo da execução do programa. Dessa forma, para serem adicionadas conforme necessário, é utilizado um algoritmo de separação para detectar se alguma delas foi violada.

Este algoritmo faz uso da sub-rotina implementada por Dezső *et al.* (2011) para encontrar a árvore de Gomory-Hu do grafo, a qual representa o corte mínimo entre todos os pares de vértices. Ainda que seja mais eficiente do que rodar o algoritmo de corte mínimo para cada par de vértices do grafo, essa sub-rotina pode ser pesada demais para ser executada diversas vezes em grafos grandes. Entretanto, como nossas instâncias são de tamanho reduzido devido a outros aspectos da dificuldade do problema, o tempo gasto para encontrar a árvore de Gomory-Hu é desprezível em relação ao tempo total. Os parâmetros da rotina *GomoryHu* são um grafo e a função de capacidade de suas arestas. O retorno é uma função m que associa, a cada aresta do grafo, as arestas do corte mínimo entre suas extremidades, com $\bar{m}$ denotando o valor do corte.

O algoritmo adiciona ao modelo a restrição de conectividade de depósitos mais violada. O funcionamento do Algoritmo 1 é mostrado em mais detalhes a seguir.

3.5.2 Restrições de Conectividade de Vértices

Similares às inequações (3.7), as restrições do tipo (3.6) garantem que a solução como um todo será conexa, desconsiderando se cada aresta é regular ou interligadora. O algoritmo

Algorithm 1: Separação das desigualdades de conectividade de depósitos.

Entrada: Grafo G; solução fracionária $\bar{x}$.
Saída: Nenhuma.

1.1 $m \leftarrow GomoryHu(G, \bar{x})$.
1.2 **para cada** $e=(i,j) \in E(G)$ **faça**
1.3 $\quad v_e \leftarrow 2(\bar{y}_i + \bar{y}_j - 1) - \bar{m}(e)$.
1.4 $e_{max} \leftarrow e \in E(G) \mid v_e$ é máximo.
1.5 **se** $v_{e_{max}} > 0$ **então**
1.6 $\quad adicionarRestrição\left(\sum_{e \in m(e_{max})} \bar{x}_e \geq 2(y_i + y_j - 1)\right)$.

utilizado é similar ao Algoritmo 1, porém adiciona todas as restrições de conectividade de vértices violadas, como mostra o Algoritmo 2.

Algorithm 2: Separação das desigualdades de conectividade de vértices.

Entrada: Grafo G; solução fracionária $\bar{x}$ e $\bar{z}$.
Saída: Nenhuma.

2.1 $m \leftarrow GomoryHu(G, \bar{x} + \bar{z})$.
2.2 **para cada** $e=(i,j) \in E(G)$ **faça**
2.3 $\quad$ **se** $\bar{m}(e) < 2$ **então**
2.4 $\quad\quad adicionarRestrição\left(\sum_{e \in m(e)} \bar{x}_e + \bar{z}_e \geq 2\right)$.

3.5.3 Desigualdades de concentrador único por ciclo

Estas são as desigualdades da forma

$$\sum_{e \in E(P)} z_e + \sum_{v \in V(P)} y_v \leq |V(P)|, \ \forall i,j \in V, \ i \neq j, \ P \in \mathcal{P}_{ij}.$$

Como são obrigatórias para a formulação estar correta, a rotina de separação deve ser capaz de encontrar uma desigualdade violada sempre que lhe for apresentada uma solução que satisfaça o restante das restrições. A rotina de separação é executada também na solução da relaxação do problema, na esperança de encontrar bons cortes cedo. Entretanto, quando as variáveis possuem valores não-inteiros, não é obrigatório encontrar um corte existente, já que esse conjunto de valores não será considerado uma solução do problema original.

É escolhido um vértice i com valor de $\bar{y}_i$ máximo (desempate aleatório) e a partir dele é feita uma busca em largura no grafo suporte G^* para encontrar caminhos a partir de i

que atinjam outros vértices com valores de y elevados. Se é encontrado um caminho cuja desigualdade correspondente é violada, esta é inserida no modelo e a busca é interrompida.

Implementamos também uma outra rotina de separação; esta, porém, não garante encontrar cortes. Sua finalidade é possibilitar mais uma tentativa de descobrir restrições violadas em situações em que a rotina anterior tenha sido infrutífera e o passo seguinte seria a ramificação. Como esta rotina é bem mais pesada, é usada somente em último caso. Nesta rotina também começamos com um vértice i com valor de $\bar{y}_i$ máximo, mas, ao contrário da anterior, usamos *backtracking* para enumerar todos os caminhos que possuem i em uma extremidade. Em cada caminho P parcial, é conhecido o valor atual d da violação da desigualdade, isto é, $d = \sum_{e \in E(P)} \bar{z}_e + \sum_{v \in V(P)} \bar{y}_v - |V(P)|$. Inicialmente, quando o caminho só contém o vértice i, a violação é $d = \bar{y}_i - 1 \leq 0$. Para evitar colocar desigualdades que tenham impacto muito pequeno, só colocamos as que possuem $d \geq 0.5$. Se, ao contrário, em algum caminho tivermos $d \leq -1$, consideramos que dificilmente um caminho gerado a partir dele terá violação suficientemente grande e, por isso, não são considerados, ou seja, esse ramo do *backtracking* é podado. Isso evita que a rotina torne-se excessivamente pesada.

3.5.4 Desigualdades de capacidade

Embora a forma apresentada

$$\sum_{e \in \delta(S)} z_e \geq 2 \frac{\sum_{i \in S} (1 - y_i)}{C - 1} - 2 \sum_{i \in S} y_i$$

da inequação (3.9) remeta mais diretamente à sua origem no artigo Lysgaard *et al.* (2004), podemos manipulá-la algebricamente para obter a forma equivalente

$$(C - 1) \sum_{e \in \delta(S)} z_e \geq 2 |S| - 2C \sum_{i \in S} y_i.$$

Computacionalmente, esta última forma tem a vantagem de ter um somatório a menos e, principalmente, evitar divisões de ponto flutuante, que seriam uma fonte de imprecisões nos cálculos. Por isso, é a versão que de fato foi implementada.

A separação das desigualdades da classe original é $\mathcal{NP}$-difícil como demonstrado por Augerat *et al.* (1995) e Naddef e Rinaldi (2001). Como a nossa versão é mais geral, em que cada vértice pode ou não ser depósito, decidimos implementar uma heurística de separação. Adaptada de Lysgaard *et al.* (2004), ela funciona da seguinte maneira. Sejam $S_1, S_2, ..., S_p$ as componentes conexas de G^*. A heurística verifica as restrições para todos os conjuntos

$S_1, S_2, ..., S_p$ e também para os conjuntos $V \setminus S_1, V \setminus S_2, ..., V \setminus S_p$, e adiciona todas as que estiverem violadas.

Note que esta desigualdade fica muito enfraquecida pelo fato de não estar definido de antemão qual o possível papel de cada vértice (cliente ou depósito). Se essa informação fosse dada na entrada, como ocorre em vários problemas da literatura, o último termo, $-2C \sum_{i \in S} y_i$, não existiria. Mais do que isso, poderia ser escrita como:

$$\sum_{e \in \delta(S)} z_e \geq 2 \left\lceil \frac{|S|}{C-1} \right\rceil .$$

Note que, além de remover os termos y_i, esta forma também possui a vantagem de arredondar a divisão para cima, fortalecendo o corte.

3.5.5 Nenhuma aresta regular entre depósitos

Uma classe de desigualdades válidas para o problema é dada por

$$z_{ij} + y_i + y_j \leq 2, \ \forall (i,j) \in E, \tag{3.30}$$

que basicamente diz que, se dois vértices i e j são depósitos ($y_i = y_j = 1$), então não pode haver uma aresta regular ($z_{ij} = 1$) entre eles. São o caso particular das desigualdades (3.8) em que os caminhos possuem apenas dois vértices.

Embora não necessária, a inserção de todas estas desigualdades logo no começo da execução do algoritmo é vantajosa, pois evita que sejam percorridos nós da árvore que seriam futuramente podados pelas restrições (3.8). Além disso, adicioná-las não pesa muito no tempo de processamento, pois trata-se de apenas uma restrição para cada aresta do grafo.

A Figura 3.3 ilustra uma solução fracionária da relaxação do problema em que uma restrição do tipo (3.30) é útil, ou seja, ela corta a solução fracionária, e a única restrição do modelo que a cortaria seria do tipo (3.8). Portanto, com isso evitamos fazer uma busca por restrições violadas desse tipo. A restrição referida ocorre na aresta em destaque (mais espessa) na figura, que chamamos de $e = (i, j)$. As variáveis possuem valores $y_i = 1$, $y_j = 1$ e $z_e = 1$. Triângulos possuem valor de y próximo a 1; arestas tracejadas possuem valor de z próximo a 1; a aresta em destaque possui $z = 1$ e extremidades com $y = 1$. Veja que esta desigualdade seria desnecessária caso os candidatos a depósito e os clientes estivessem separados na entrada do problema, pois saberíamos que $z_e = 0$ para toda aresta e que ligasse dois depósitos.

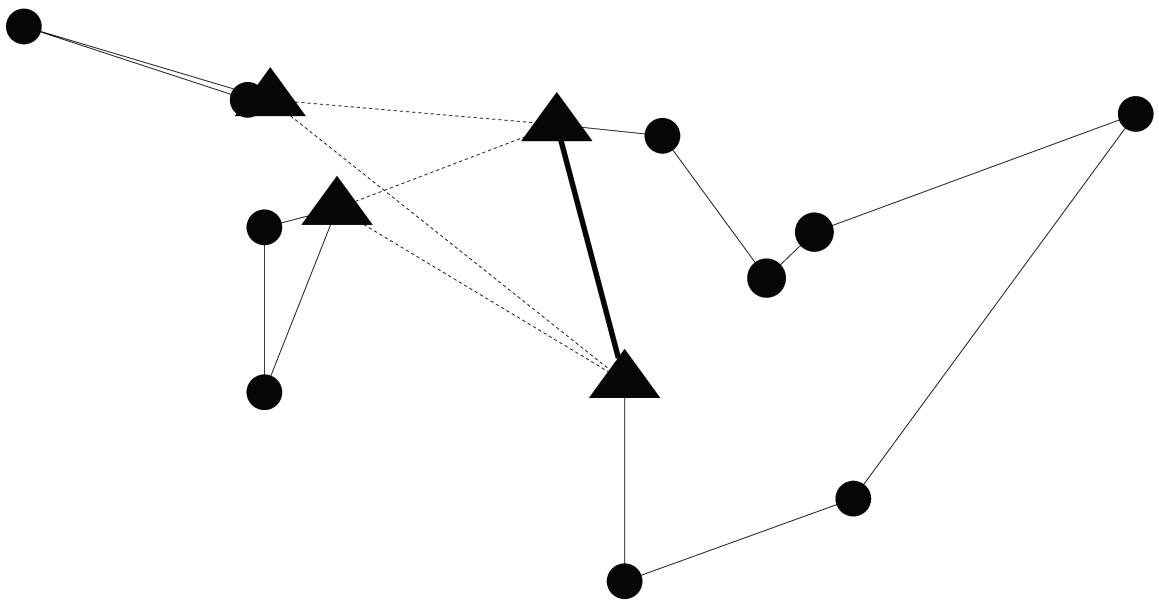


Figura 3.3: Exemplo de solução fracionária cortada por uma restrição (3.30).

3.5.6 Arestas interligadoras somente entre depósitos

Tal qual a classe anterior de inequações, esta também é desnecessária para o modelo, mas pode ajudar a agilizar a busca. A restrição é dada pelas inequações

$$x_{ij} \leq y_i,\ \forall (i,j) \in E \tag{3.31}$$

e

$$x_{ij} \leq y_j,\ \forall (i,j) \in E \tag{3.32}$$

e garante que uma aresta só poderá ser interligadora se suas duas extremidades forem depósitos. A inclusão destas desigualdades também não é muito cara: apenas duas restrições para cada aresta do grafo. A Figura 3.4 ilustra a situação a ser evitada. Note que há uma aresta regular entre os depósitos h_1 e h_2. Esta é mais uma restrição que seria desnecessária caso soubéssemos o papel de cada vértice, pois poderíamos fazer $x_e = 0$ para toda aresta e que incidisse em um cliente.

3.5.7 Desigualdades de cardinalidade de caminhos

A desigualdade de cardinalidade de caminhos limita o número de vértices (e arestas) dos ciclos regulares. É dada por:

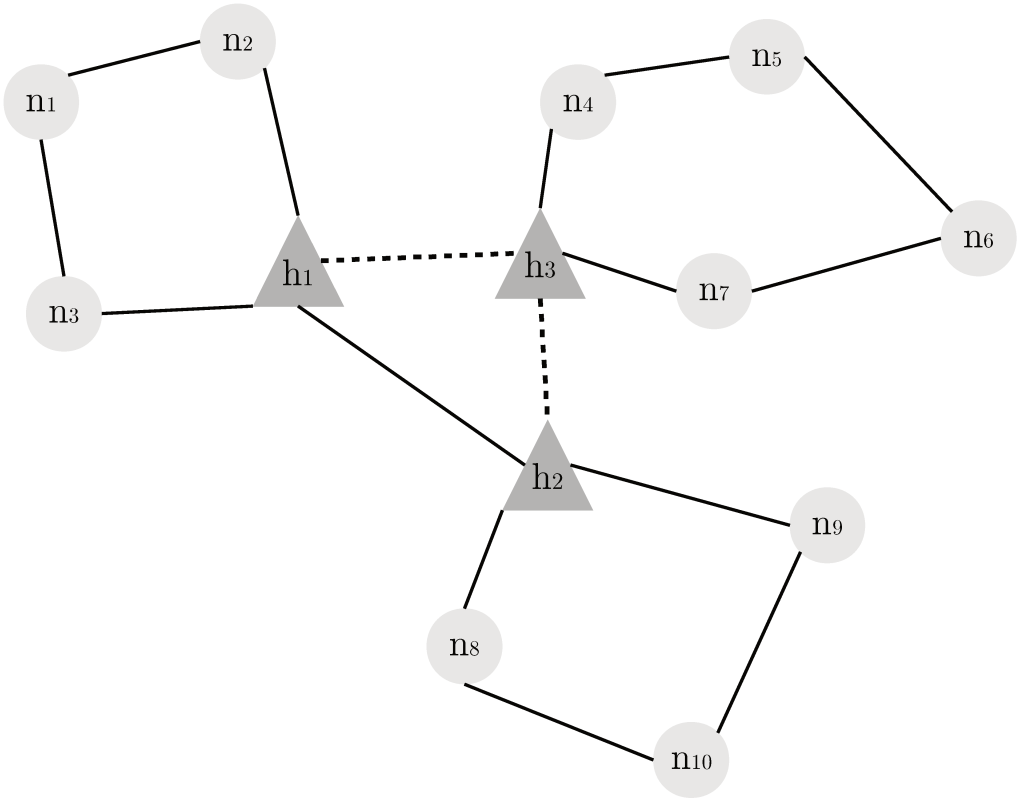


Figura 3.4: Ilustração de situação evitada pelas restrições (3.31) e (3.32).

$$\sum_{e \in E(P)} z_e \leq C - 1, \ \forall P \in \mathcal{P}, |E(P)| = C. \tag{3.33}$$

A inequação (3.33) é a desigualdade *Cardinality-Path* utilizada por Bauer *et al.* (2002) no Problema de Ciclos com Restrições de Cardinalidade. Se um caminho com exatamente C arestas estivesse na solução, ele precisaria de no mínimo mais uma aresta para fechar o circuito, totalizando $C + 1$ arestas, o que não é permitido. Logo, podemos colocar essa desigualdade para eliminar esse caminho.

Note que essa desigualdade só é colocada quando o caminho possui exatamente C arestas. Se ele for mais curto, ela é redundante, já que afirmaria que a soma de $C - 1$ (ou menos) variáveis binárias é no máximo $C - 1$, o que é óbvio. Se o caminho for mais longo, essa desigualdade não é válida, uma vez que pode ser que, na solução ótima, uma (ou mais) das arestas do caminho considerado não esteja presente e, portanto, na verdade sejam dois (ou mais) caminhos disjuntos, o que faria com que o limite de arestas fosse maior.

Para encontrar estas desigualdades violadas, aproveitamos a busca descrita na seção 3.5.3, que é realizada na separação das desigualdades do tipo (3.8), e, caso o caminho entre o vértice

inicial e o atual tenha exatamente C arestas, verificamos se ele satisfaz a desigualdade de cardinalidade.

3.6 Heurística

Foi implementada uma heurística para buscar boas soluções viáveis para o problema de maneira substancialmente mais rápida do que um algoritmo exato. Essas soluções servem tanto para acelerar o algoritmo exato, quanto para obter alguma boa solução nas instâncias que são grandes a ponto de não serem tratáveis por ele.

A aceleração ocorre de duas maneiras: o valor dessa solução é informado para o algoritmo exato, o que lhe permite descartar rapidamente ramos da árvore de busca que sejam certamente infrutíferos; e as variáveis do programa linear são inicializadas com os valores correspondentes aos da solução obtida na heurística, o que permite ao algoritmo exato iniciar com uma solução viável.

3.6.1 Representação da solução

Uma solução viável x é representada por uma tupla $t_x = \langle r_1, r_2, \ldots, r_{|t_x|} \rangle$, com cada r_i sendo a sequência de vértices formando um ciclo regular, e o primeiro vértice de cada r_i corresponde ao seu concentrador. O ciclo interligador é obtido tomando o primeiro vértice de cada elemento de t_x. A Figura 3.5 ilustra uma solução viável e sua tupla correspondente.

3.6.2 Solução Inicial

A estratégia que utilizamos consiste em obter uma solução viável aleatória. Naturalmente, queremos que toda solução viável possa ser obtida por este algoritmo. É necessário também ter cuidado para evitar soluções com ciclos degenerados ou que extrapolem a capacidade. Descrevemos a seguir uma visão geral do funcionamento do algoritmo.

Primeiro escolhemos aleatoriamente três vértices para cada ciclo regular. Isso garante que não haverá ciclos degenerados. Depois, para cada vértice ainda não atribuído, escolhemos aleatoriamente um ciclo que ainda não tenha C vértices, garantindo que nenhum ciclo extrapolará a capacidade. A ordem dos vértices em cada ciclo é a ordem de inserção, com o depósito sendo o primeiro. A ordem de cada depósito no ciclo interligador é a ordem dos ciclos.

Toda solução viável pode ser gerada por esse procedimento, e todas as soluções por ele geradas são viáveis. É um algoritmo bem rápido e satisfatório para o que se propõe.

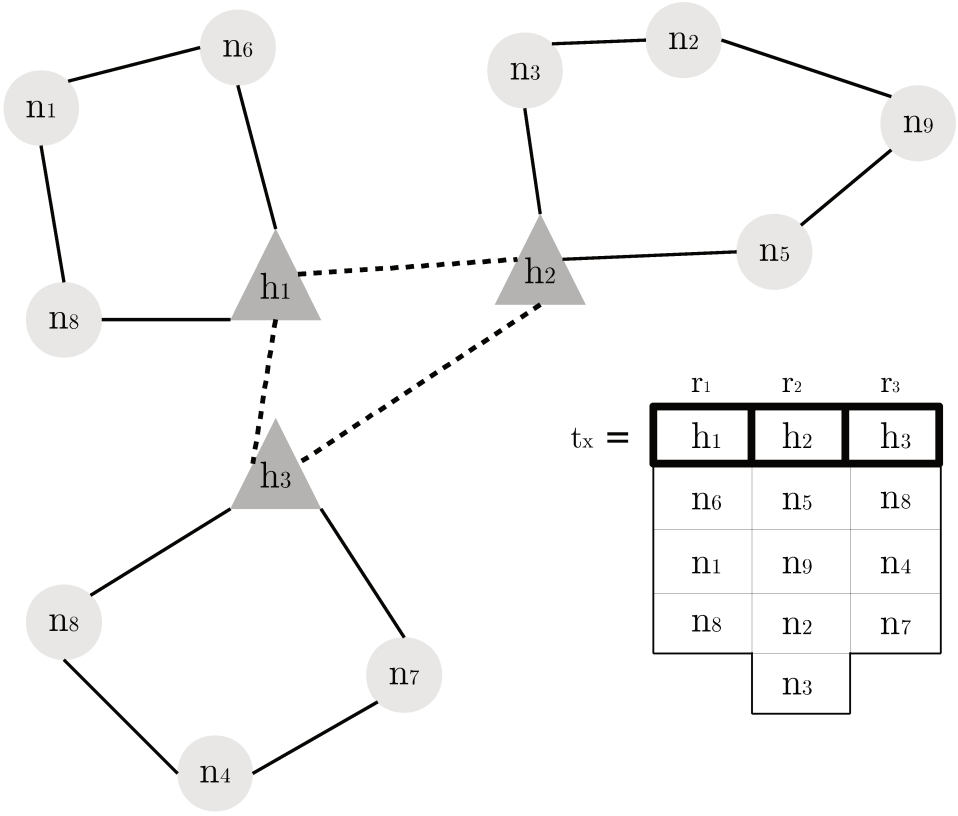


Figura 3.5: Tupla representando uma solução viável.

Algorithm 3: Solução inicial aleatória.

Entrada: Instância I, incluindo $G = (V, E)$, K e C.
Saída: Solução x.

3.1 $V' \leftarrow embaralhar(V)$.
3.2 **para cada** $i \in \{1, 2, \ldots, K\}$ **faça**
3.3 $\quad r_i \leftarrow V'_i$.
3.4 **para cada** $i \in \{K+1, K+2, \ldots, |V|\}$ **faça**
3.5 $\quad s \leftarrow sortear\,(\{j \in \{1, 2, \ldots, K\} \mid C > |r_j|\})$.
3.6 $\quad r_s \leftarrow (r_s, V'_i)$.
3.7 **retorna** r.

3.6.3 Busca Local

Definimos três operadores para encontrar soluções vizinhas de uma dada solução viável: *Mover vértices*, *Trocar vértices* e *Escolher Concentradores*. São baseados nas operações descritas em Aarts e Lenstra (1997). Além deles, é usada também a heurística de *Lin-Kernighan* para melhorar a ordem dos clientes dentro de um mesmo ciclo. A seguir, vemos em detalhes como funciona cada uma das operações, considerando que, para uma solução viável x, aplicamos os

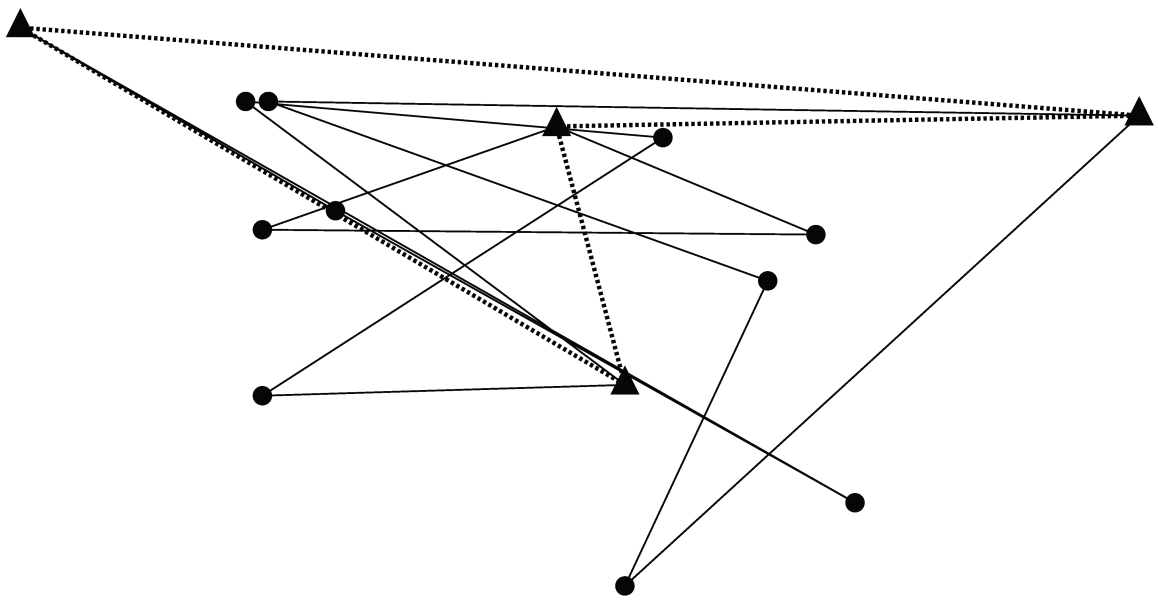


Figura 3.6: Solução viável inicial aleatória.

operadores sobre uma estrutura dada por $R = \left\{r_1 = t_x\left(1\right), r_2 = t_x\left(2\right), \ldots, r_{|t_x|} = t_x\left(|t_x|\right)\right\}$. Consideramos que a ordem dos ciclos importa, ou seja, (r_k, r_s) é diferente de (r_s, r_k).

O operador OP_1 (Mover vértices) remove um vértice de um ciclo regular r_k e insere-o em um outro ciclo regular r_s. Em outras palavras, dado um par de ciclos (r_k, r_s), para cada $i \in r_k$ e cada aresta $\{j, u\}$ do ciclo r_s, remova i de r_k e adicione em r_s de modo que a aresta $\{j, u\}$ torna-se as arestas $\{j, i\}$ e $\{i, u\}$. Cada potencial operação bem-sucedida (ou seja, uma operação que diminui o custo da solução) é guardada em uma lista L. Como a operação é testada para todos os pares de ciclos, a lista L contém todas as possíveis operações que seriam bem-sucedidas se aplicadas. Por fim, é realizada uma operação qualquer da lista L sorteada uniformemente, ao invés de considerar a que mais diminui o custo da solução. O Algoritmo 4 descreve o operador OP_1, e a figura 3.7 ilustra a operação. Nesse caso, o vértice i é removido do ciclo r_k e inserido no ciclo r_s entre os vértices j e u.

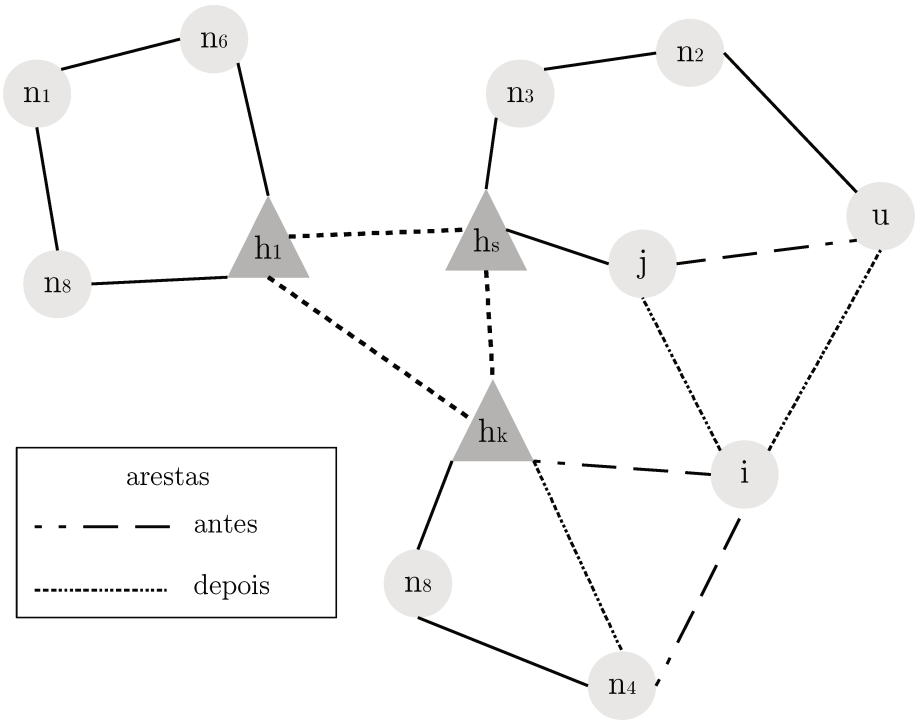


Figura 3.7: Ilustração da heurística *Mover Vértices*.

Algorithm 4: OP_1 – Mover Vértices

Entrada: Instância I; solução x.
Saída: Solução $\bar{x}$.

```
4.1  x̄ ← x; L ← ∅.
4.2  para cada par de ciclos regulares {r_k, r_s} ∈ x̄ faça
4.3      para cada vértice i ∈ r_k faça
4.4          para cada aresta {j, u} ∈ r_s faça
4.5              opr ← remover i de r_k e {j, u} de r_s e inserir arestas {j, i} e {i, u} em r_s.
4.6              Aplicar opr a x̄.
4.7              se F(x̄) < F(x) então
4.8                  L ← L ∪ {opr}.
4.9              Desfazer opr em x̄.
4.10 opr ← selecionar um elemento de L de maneira uniformemente aleatória.
4.11 Aplicar opr a x̄.
4.12 retorna x̄.
```

No operador OP_2 temos a operação de troca de vértices. Para cada par de ciclos regulares r_k e r_s e cada vértices $i \in r_k$ e $j \in r_s$, i é movido para r_s e j, para r_k. Tal qual o OP_1, este operador também é considerado para todos os pares de ciclos, e cada operação que melhoraria o valor da solução viável atual é guardada em uma lista L, sendo escolhida aleatoriamente apenas uma operação de L e imediatamente aplicada. O algoritmo é igual ao apresentado no Algoritmo 4, com exceção da linha 4.4, em que o OP_2 considera para cada vértice $j \in r_s$, e da linha 4.5, em que *opr* consiste em trocar i e j como na Figura 3.8. Ambos os operadores são aplicados somente a vértices clientes.

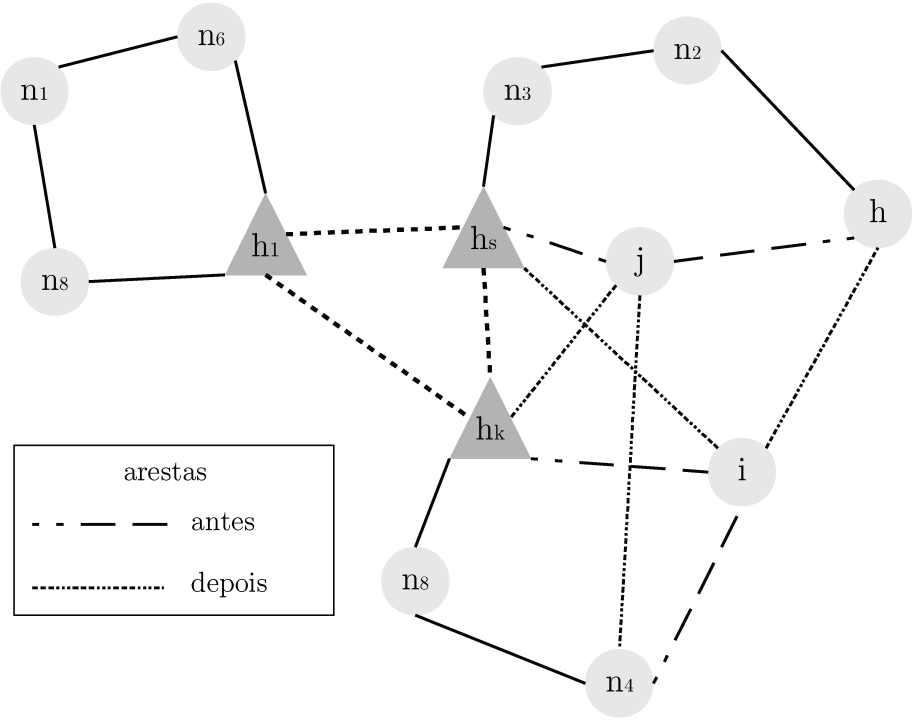


Figura 3.8: Ilustração da heurística *Trocar Vértices.*

O terceiro operador, OP_3, ao contrário dos anteriores, atua sobre os concentradores. Em cada ciclo regular, ele busca o vértice que mais reduz o custo da solução caso seja convertido em concentrador. Em outras palavras, para cada ciclo regular r_k e cada $i \in r_k$, considere agora i como o concentrador e atualize as arestas para manter a viabilidade da solução. Se essa mudança reduzir o custo da solução, então é uma operação considerada bem-sucedida. Novamente, as operações bem-sucedidas são armazenadas em uma lista L e apenas uma delas, escolhida de modo uniformemente aleatório, é de fato aplicada. A Figura 3.9 mostra o vértice i tornando-se o novo concentrador do ciclo r_k, com a descrição em detalhes dada pelo Algoritmo 5. Note que este operador diversifica a busca ao considerar novas soluções viáveis caracterizadas por diferentes concentradores.

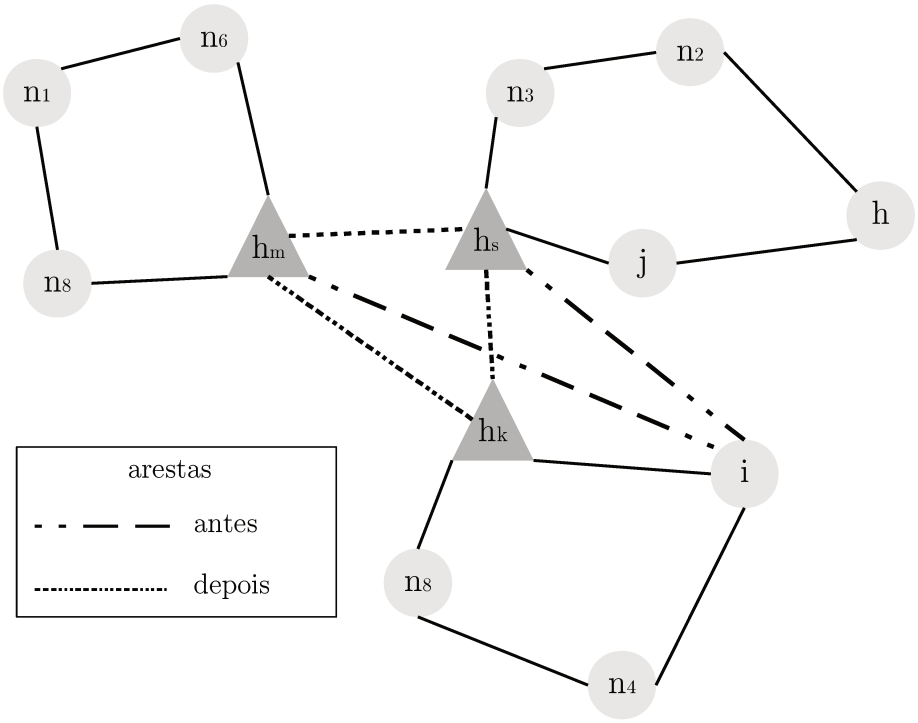


Figura 3.9: Ilustração da heurística *Escolher Concentradores.*

Algorithm 5: OP_3 – Escolher Concentradores

Entrada: Instância I; solução x.
Saída: Solução $\bar{x}$.

5.1 $\bar{x} \leftarrow x$; $L \leftarrow \emptyset$.
5.2 **para cada** *ciclo regular* $r_k \in \bar{x}$ **faça**
5.3 Seja h_k o concentrador de r_k.
5.4 **para cada** *vértice* $i \in r_k$ *com* $i \neq h_k$ **faça**
5.5 Sejam $e_1 = \{h_m, h_k\}$ e $e_2 = \{h_s, h_k\}$ as arestas interligadoras incidentes em h_k.
5.6 *opr* $\leftarrow$ *trocar* arestas e_1 e e_2 por $\{h_m, i\}$ e $\{h_s, i\}$ em $\bar{x}$.
5.7 Aplicar *opr* a $\bar{x}$.
5.8 **se** $\mathcal{F}(\bar{x}) < \mathcal{F}(x)$ **então**
5.9 $L \leftarrow L \cup \{opr\}$.
5.10 Desfazer *opr* em $\bar{x}$.
5.11 *opr* $\leftarrow$ selecionar um elemento de L de maneira uniformemente aleatória.
5.12 Aplicar *opr* a $\bar{x}$.
5.13 **retorna** $\bar{x}$.

A heurística de *Lin-Kernighan* é normalmente utilizada para o Problema do Caixeiro Viajante e é descrita por Applegate *et al.* (2003). Essa heurística é baseada em λ-*moves*, cujos passos consistem em remover λ arestas e remontar os caminhos resultantes – possivelmente invertendo alguns deles – de modo a obter um novo ciclo, mais curto (ou mais barato) que o original. Casos especiais para $\lambda = 2$ e para $\lambda = 3$ são comuns, mas a heurística de Lin-Kernighan adapta o valor de λ de modo crescente durante a execução, observando se pode melhorar o custa do ciclo. Uma implementação eficiente dessa heurística e algumas variantes são discutidas por Helsgaun (2000). A Figura 3.10 ilustra o resultado da aplicação da heurística de Lin-Kernighan em uma solução viável qualquer.

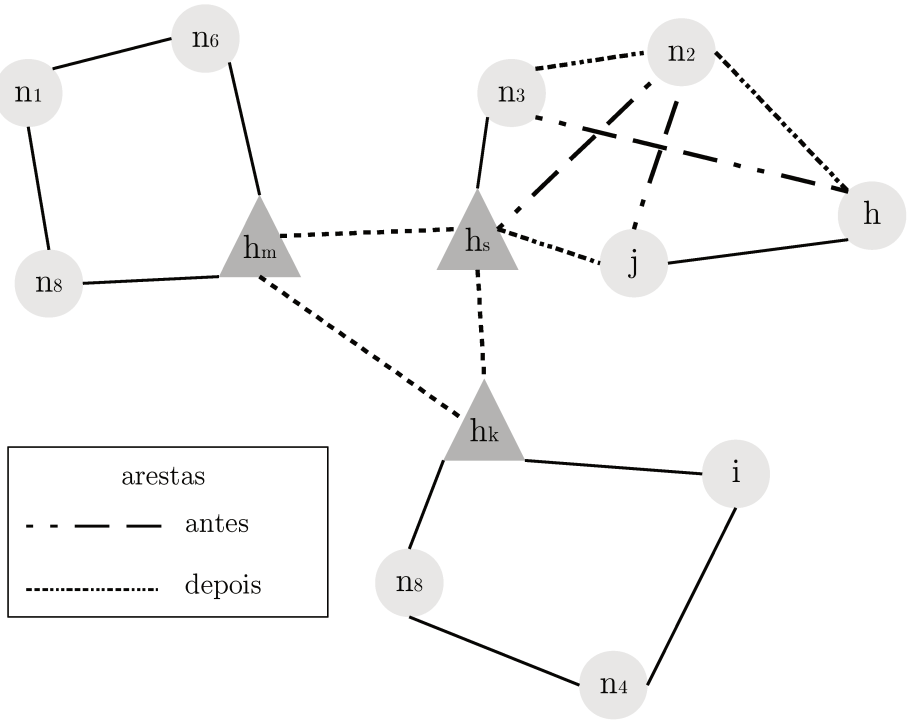


Figura 3.10: Ilustração da heurística de *Lin-Kernighan.*

O algoritmo de busca local que implementamos consiste em aplicar os operadores OP_1, OP_2 e OP_3 ciclicamente, até nenhum deles conseguir mais melhorar a solução. Nesse momento, aplica-se a heurística de Lin-Kernighan, que é deixada por último por ser consideravelmente mais lenta de executar. Quando a Lin-Kernighan melhora as rotas de uma solução x, é natural tentar também melhorar a localização dos concentradores. Portanto, é executado várias vezes o operador OP_3, em uma operação que chamamos de *Multi-*OP_3, cujo propósito é percorrer o ciclo interligador de x até não se conseguir mais melhorar o valor da solução. Para um ciclo r_k (inicialmente $k = 1$), experimenta-se cada $i \in r_k$ como concentrador. Se houver tal $i \in r_k$ que melhore o valor da solução, então o *Multi-*OP_3 é reiniciado para o ciclo

r_s, com $s = k + 1$; caso contrário, continua-se a analisar o ciclo seguinte. Se a operação começou em k, deve ser parada após analisar todos os ciclos regulares e não encontrar um novo concentrador i que melhore o custo da solução x.

Se alguma das operações Lin-Kernighan ou *Multi-OP*$_3$ for bem-sucedida, a busca local volta à etapa de aplicação dos operadores. Caso contrário, temos um mínimo local para a vizinhança definida pelas operações implementadas e esse é o resultado final da busca local. O Algoritmo 6 descreve essa rotina em mais detalhes.

Algorithm 6: Busca local

Entrada: Instância I; solução x.
Saída: Solução $\bar{x}$.

```
6.1  x̄ ← x.
6.2  repita
6.3      out ← falso.
6.4      repita
6.5          in ← falso.
6.6          x̄ ← OP1(I, x̄); x̄ ← OP2(I, x̄); x̄ ← OP3(I, x̄).
6.7          se F(x̄) < F(x) então
6.8              in ← verdadeiro; x ← x̄.
6.9      até in = verdadeiro
6.10     x̄ ← LinKernighan(I, x̄). x̄ ← Multi-OP3(I, x̄).
6.11     se F(x̄) < F(x) então
6.12         out ← verdadeiro; x ← x̄.
6.13 até out = verdadeiro
6.14 retorna x̄.
```

Partindo da solução viável aleatória inicial da Figura 3.6 e aplicando as operações descritas acima, obtém-se a solução final da execução da Busca Local, dada a seguir pela Figura 3.11. A heurística que implementamos começa gerando uma solução viável inicial. Uma vez gerada, executa-se a busca local descrita acima. O procedimento todo é então refeito a partir de outra solução viável inicial. Há um parâmetro MAX indicando o número máximo de iterações *consecutivas* sem melhoras. Em outras palavras, a cada vez que uma solução encontrada é melhor do que todas as outras já conhecidas, a contagem do número de iterações é zerada. Quando esse limite é atingido, a melhor solução encontrada é então considerada o resultado final de nossa heurística. Os passos realizados são mostrados mais claramente no Algoritmo 7.

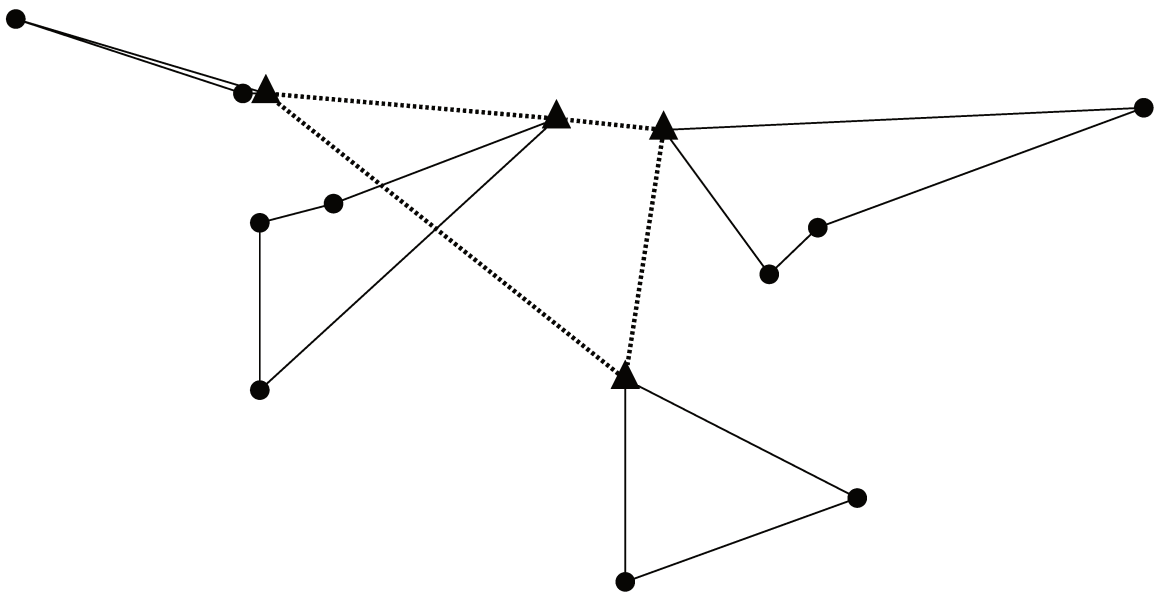


Figura 3.11: Solução final encontrada pela Busca Local a partir da solução viável inicial da Figura 3.6.

Algorithm 7: Heurística

Entrada: Instância I; número máximo de iterações MAX.
Saída: Solução para o LRCMM.

7.1 $iters \leftarrow 1$.
7.2 **enquanto** $iters \leq MAX$ **faça**
7.3 $x \leftarrow$ construa uma solução viável inicial para I.
7.4 $\bar{x} \leftarrow$ `BuscaLocal`(I, x).
7.5 **se** $\mathcal{F}(\bar{x}) < \mathcal{F}(x)$ **então**
7.6 $iters \leftarrow 0$; $x \leftarrow \bar{x}$.
7.7 **senão**
7.8 $iters \leftarrow iters + 1$.
7.9 **retorna** x.

Capítulo 4

Experimentos Computacionais

4.1 Informações gerais

4.1.1 Ambiente computacional

Foi implementado um algoritmo exato, isto é, um algoritmo que garante encontrar a solução ótima. Para tanto, foi utilizada a técnica de *Branch-and-cut*, descrita na seção 2.1.5. Inicialmente utilizamos a versão acadêmica da biblioteca comercial ILOG CPLEX, da IBM (2009) e, posteriormente, a biblioteca Gurobi (2013). A última apresentou, consistentemente, desempenho muito superior para o nosso problema e, portanto, foi escolhida. Ambas fornecem um algoritmo básico de *branch-and-cut* que pode ser customizado com diversas opções. O Gurobi, por não ser tão maduro, não possui algumas das funcionalidades do CPLEX, o que em alguns casos foi uma dificuldade a mais. Por exemplo, não é oferecida a possibilidade de ramificar (*branch*) em uma soma de variáveis, apenas em uma única variável por vez.

Para a representação interna do grafo, utilizamos a biblioteca LEMON, desenvolvida por Dezső *et al.* (2011) e pertencente ao projeto COIN-OR, de Lougee-Heimer (2003). Ela facilita o uso de vários algoritmos em grafos, como buscas, cortes e fluxos. Especialmente útil foi o algoritmo de Gomory-Hu, descrito por Gomory e Hu (1961), para encontrar o corte mínimo entre todos os pares de vértices.

Os experimentos computacionais foram realizados em uma máquina com processador Intel® Xeon® CPU X3430 (64 bits, 4 núcleos, 2.40 GHz) e 8 GB de memória RAM. O sistema operacional instalado nessa máquina era Ubuntu Linux 12.04.3 com *kernel* 3.2.0. A biblioteca de programação linear usada foi Gurobi Optimizer 5.6.2, enquanto que a de manipulação de grafos foi a LEMON 1.3. Nosso executável foi gerado com o compilador GCC 4.8.2.

4.1.2 Ordem das rotinas de separação

Quando o algoritmo de *Branch-and-cut* obtém a solução da relaxação do problema, é invocada uma função que escolhe quais rotinas de separação serão executadas. Nessa função, somos obrigados a executar todas as rotinas de separação das subseções 3.5.1, 3.5.2 e 3.5.4 (ou pelo menos até uma delas ter encontrado uma restrição violada), pois estas encontram restrições que fazem parte do modelo. Se não o fizermos, corremos o risco de aceitar soluções inválidas. Além delas, opcionalmente podemos executar outras rotinas para procurar bons cortes que agilizem a resolução do problema.

É preciso, portanto, definir em que ordem essas rotinas são chamadas. Isso pode interferir no desempenho total porque as rotinas de separação possuem tempos de execução muito diferentes. Como em vários casos é encontrada uma restrição violada na primeira rotina executada e isso permite evitar a execução das outras rotinas, é interessante executar primeiro as rotinas mais rápidas, na esperança de não executar as mais pesadas e evitar tornar o modelo pesado demais por inserir muitos cortes. Por outro lado, algumas soluções violam restrições de diversos tipos, as quais descartam diferentes porções do espaço de busca. Por isso, pode ser vantajoso executar várias rotinas em sequência, independentemente do resultado obtido por cada uma. Isso tem o objetivo de evitar que o programa alterne entre a resolução da relaxação fracionária e a inserção de cortes da primeira rotina de separação, cada corte possivelmente conseguindo um avanço muito pequeno em relação ao anterior. Ao chamarmos outras rotinas de separação independentemente do sucesso das primeiras, garantimos a inserção de cortes de variados tipos e, com sorte, suficientemente diferentes para evitar esse problema.

As melhores rotinas de separação em um critério podem ou não ser as melhores no outro. Dessa forma, variamos a ordem delas em alguns experimentos e chegamos à ordem que teve melhores resultados: busca em largura (seção 3.5.3); restrições de conectividade de vértices (seção 3.5.2); de conectividade de depósitos (3.5.1); de capacidade (3.5.4); e, somente caso as anteriores não tenham sido bem-sucedidas, entra em ação a rotina de enumeração de caminhos (seções 3.5.3 e 3.5.7).

Vale lembrar que todos os cortes encontrados por essas rotinas devem ser válidos para toda solução viável do problema. Além disso, elas podem ser invocadas em soluções da relaxação do problema, ou seja, com valores fracionários nas variáveis, o que dificulta a elaboração de rotinas corretas.

4.1.3 Prioridade de ramificação

Pacotes de otimização que oferecem facilidades para a implementação do *Branch-and-cut*, como CPLEX e Gurobi, permitem que o usuário especifique a prioridade de cada variável.

Quando o algoritmo está em um nó fracionário e não consegue inserir novos cortes que eliminem aquela solução da relaxação, ele realiza uma ramificação (*branch*) na árvore de busca, criando dois nós (descendentes do ramificado) que formam uma partição do espaço de busca.

Enquanto o CPLEX permite ramificar de acordo com uma variável ou uma expressão linear, o Gurobi oferece apenas a primeira alternativa. Então, um exemplo de ramificação seria fazer $x_e = 0$ em um ramo e $x_e = 1$ no outro. No CPLEX, poderíamos, por exemplo, dividir em $x_e + x_{e'} = 0$ e $x_e + x_{e'} \geq 1$.

Após alguns experimentos, concluímos que é mais vantajoso dar maior prioridade para as variáveis do tipo y, usadas para determinar se um vértice pertence ou não ao ciclo interligador. Uma explicação possível para esse fato é que definir rapidamente os depósitos simplifica consideravelmente o problema, pois determinar as variáveis do tipo x passa a ser questão de resolver um TSP no ciclo formado pelos depósitos (que, embora seja por si só um problema difícil, é resolvido em um grafo substancialmente menor que o original). Com as variáveis dessas duas classes definidas, o problema fica imensamente simplificado.

4.2 Instâncias

Para nossos testes experimentais, utilizamos o conjunto de instâncias para o Problema do Caixeiro Viajante chamada TSPLIB, definida por Reinelt (1991). Ela contém grafos não-orientados originalmente para serem usados em resolvedores do Problema do Caixeiro Viajante. Os grafos em geral são inspirados em casos reais, como mapas de países e placas de circuitos integrados. Escolhemos todos os que possuem no máximo 100 vértices, totalizando 28 grafos. Esse tamanho de grafo foi escolhido pois já é bastante difícil, dada a natureza do problema, e está de acordo com outros artigos relacionados. Rodríguez-Martín *et al.* (2014), por exemplo, resolveram a maioria das instâncias com até 50 vértices, porém com o dobro do tempo limite que utilizamos. Portanto, o tamanho máximo escolhido (100 vértices) é suficientemente grande para fazermos uma boa análise dos resultados.

Na adaptação para o nosso problema, não precisamos fazer qualquer alteração nos grafos em si. Foi necessário, entretanto, criar valores para os parâmetros k, C e α. O valor de α é livre, isto é, qualquer valor permite soluções viáveis. Optamos por utilizar $\alpha \in \{0{,}2; 0{,}4; 0{,}6; 0{,}8\}$, com valores $\alpha < 1$ em concordância com a literatura – veja O'Kelly (1987), Contreras *et al.* (2010) e de Camargo *et al.* (2013). Esse desconto no transporte entre concentradores deve-se ao grande volume de passageiros dessas ligações, permitindo que sejam realizadas viagens com menos assentos vagos, aumentando a eficiência do transporte e, portanto, reduzindo os custos.

Já os valores de k e C precisam ser escolhidos com mais cuidado para não causar inviabilidade:

- $C \geq 3$, pois os ciclos regulares são não-degenerados;
- $k \geq 3$, pois o ciclo interligador é não-degenerado;
- $k \leq \frac{n}{3}$, para o caso em que os ciclos regulares são não-degenerados; $k \leq n$, caso contrário;
- $kC \geq n$, pois os ciclos regulares cobrem o grafo todo.

O artigo de Andrade *et al.* (2013) utiliza cinco combinações de valores de k e C para cada grafo. Entretanto, três delas não são possíveis na nossa versão sem ciclos degenerados, pois necessitam de ciclos degenerados para ter soluções viáveis. Por isso, realizamos experimentos com os dois cenários restantes, ST e SL, na nomenclatura utilizada naquele artigo.

Para um dado valor de n, os dois grupos apresentam o mesmo valor de k, mas o cenário ST é mais apertado que o SL, ou seja, possui um valor menor de C. Consequentemente, todas as soluções viáveis do cenário ST também são soluções viáveis de SL. Mais precisamente, os cenários são definidos da seguinte forma:

- Cenário ST: $k = \left\lceil \frac{n}{5} \right\rceil$, $C = \left\lceil \frac{n}{k} \right\rceil$.
- Cenário SL: $k = \left\lceil \frac{n}{5} \right\rceil$, $C = \left\lfloor \frac{9}{5} \left\lceil \frac{n}{k} \right\rceil \right\rfloor$.

Note que o cenário ST faz com que os ciclos tenham o mínimo de folga possível, diferentemente do cenário SL.

Existindo dois cenários, quatro valores de α e 28 grafos, há um total de $2 \cdot 4 \cdot 28 = 224$ casos de teste.

4.3 Resultados

4.3.1 Heurística

O valor da melhor solução conhecida pela heurística tende a decrescer rapidamente nas primeiras iterações e depois diminuir a velocidade, até ficar praticamente estagnado. Nesse ponto, ou temos a solução ótima, ou temos uma solução boa a ponto de ser raro encontrarmos uma outra solução melhor em um número aceitável de iterações. Portanto, queremos que a heurística faça um número de iterações suficientemente grande para chegarmos a esse ponto de estagnação, mas não muito mais que isso, para não gastar tempo em vão.

Ao invés de fixarmos o número absoluto de iterações, julgamos mais razoável estipular o número máximo de iterações consecutivas que a heurística pode fazer sem melhorias. A partir desse ponto, interrompe-se sua execução e retorna-se a melhor solução conhecida. Ajustando esse parâmetro, pode-se facilmente regular a heurística para obter melhores resultados ao custo de maior tempo de execução. Em nossos testes, chegamos à conclusão que fixar esse parâmetro em 16000 é suficiente para todas as instâncias testadas.

As Figuras 4.1 e 4.2 ilustram, para duas instâncias diferentes, o comportamento do valor da melhor solução conhecida em função do total de iterações executadas pela heurística a cada instante. Na primeira, a execução foi interrompida na iteração 35321, após ter melhorado o valor nas iterações 12394 e 19337. Na segunda, as duas últimas melhorias ocorreram nas iterações 5415 e 20341, depois executando até a iteração 36321.

Em ambos os casos, a execução encerrou-se porque as últimas 16000 iterações foram infrutíferas, como exibido nos gráficos, e diminuir o parâmetro nos faria perder uma melhoria da solução por volta da iteração 20000.

Com essa calibração da heurística, executamos a heurística para todos os casos estudados. Os resultados estão nas tabelas da seção a seguir, junto com os resultados do algoritmo exato para facilitar a visualização.

4.3.2 Algoritmo Exato

As tabelas de A.1 a A.16 com todos os resultados experimentais obtidos encontram-se no Apêndice. Para facilitar a leitura do texto, apresentamos aqui apenas a Tabela 4.1, que sumariza os resultados das outras tabelas. Cada linha desta tabela apresenta, para um dado conjunto de parâmetros (cenário, versão do problema e valor de α), as médias dos resultados dos 28 grafos utilizados como entrada do programa, igual à última linha de cada tabela.

Cada linha das tabelas do Apêndice consiste das seções de entrada, heurística e algoritmo exato. A entrada contém o nome da instância, o número de vértices n, o número de depósitos K, e a cardinalidade máxima C de cada ciclo regular. A seção da heurística possui o tempo de execução (em segundos) e o valor da solução encontrada. Já a seção do algoritmo exato *Branch-and-cut* apresenta o número de cortes inseridos, o número de nós explorados pela árvore de busca, o tempo total de execução, o valor da solução encontrada, e o *gap* entre os limitantes superior e inferior encontrados. Além disso, há uma coluna indicando a razão entre o valor da melhor solução da heurística e o da melhor solução encontrada pelo exato.

Nas instâncias em que o algoritmo exato consegue provar que encontrou uma solução ótima, há um asterisco ($*$) indicando esse fato; caso contrário, há apenas o valor da melhor solução conhecida. Se o valor da solução encontrada pela heurística mostrou-se ótimo, esse fato é indicado por um sinal de adição ($+$).

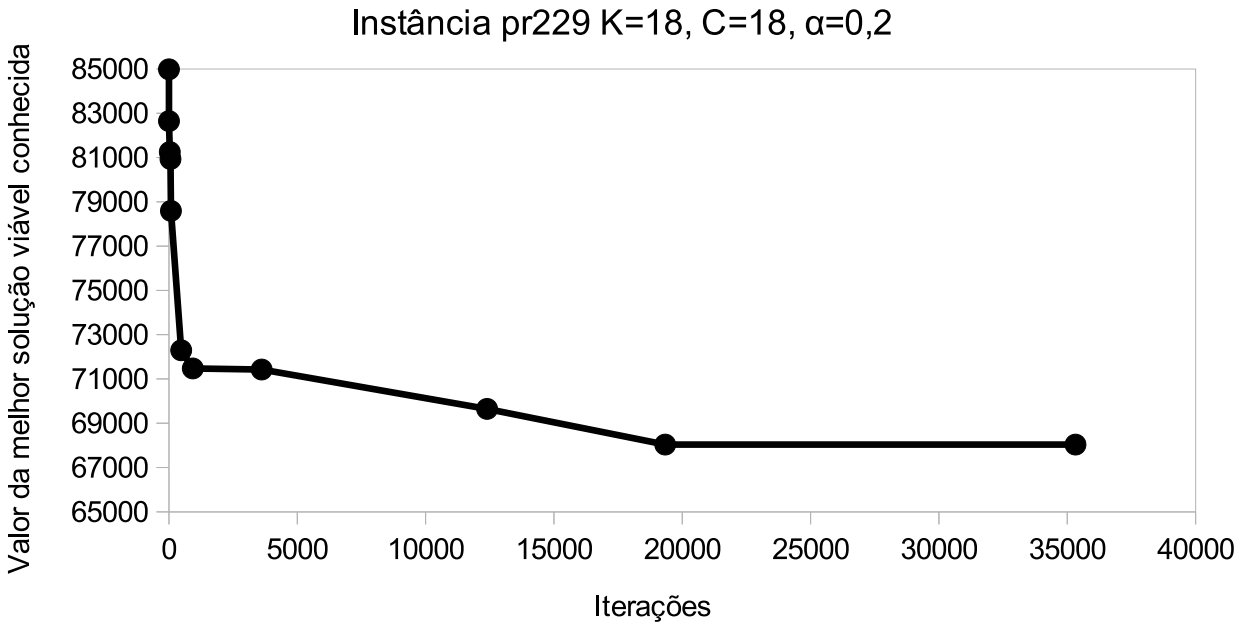


Figura 4.1: Comportamento do valor da heurística em função do número de iterações.

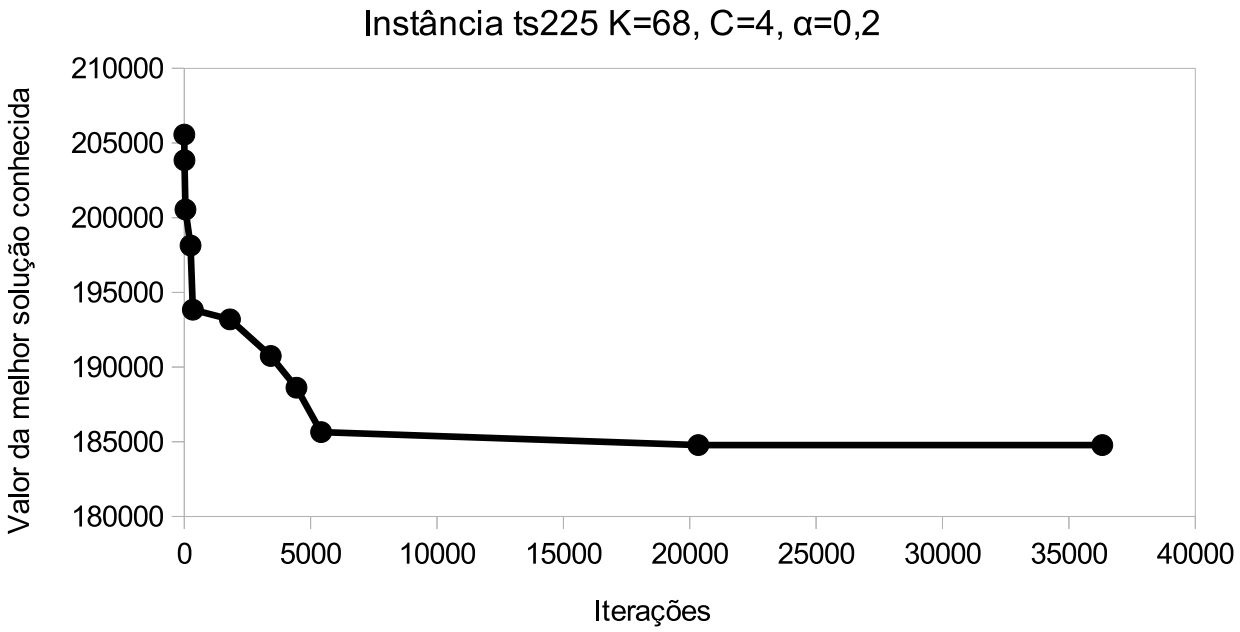


Figura 4.2: Comportamento do valor da heurística em função do número de iterações.

Tabela 4.1: Médias dos resultados das 28 instâncias para cada conjunto de parâmetros.

Cenário	α	Ciclos degenerados	Heurística	Branch-and-cut				Razão
			tempo (s)	cortes	nós	tempo (s)	Gap (%)	
ST	2	Não	116,05	24697,68	15220,25	2713,94	9,85	1,00
ST	4	Não	118,34	41191,68	20485,54	2723,36	11,41	1,00
ST	6	Não	112,23	47732,39	23637,64	2735,86	12,56	1,00
ST	8	Não	128,74	65742,93	24202,93	2737,70	13,23	1,00
ST	2	Sim	104,47	87713,82	26366,43	3226,63	24,47	1,00
ST	4	Sim	108,74	82659,43	25397,93	3221,82	23,62	1,00
ST	6	Sim	137,29	82804,71	27010,25	3222,66	23,37	1,00
ST	8	Sim	114,96	80005,14	27349,29	3222,14	23,67	1,00
SL	2	Não	279,34	12879,36	9675,21	2104,50	4,34	1,01
SL	4	Não	278,39	21491,68	13143,32	2451,93	5,72	1,01
SL	6	Não	305,61	30880,89	13637,11	2449,16	7,24	1,00
SL	8	Não	316,68	44813,54	14674,50	2450,21	8,59	1,00
SL	2	Sim	173,07	57873,82	25925,18	3217,42	20,22	1,00
SL	4	Sim	205,94	46847,86	20370,43	3094,75	18,86	1,00
SL	6	Sim	231,70	45441,50	21809,46	3095,89	18,30	1,00
SL	8	Sim	247,19	39064,36	23007,75	3094,55	17,71	1,00

4.3.2.1 Versão sem ciclos degenerados

Para a versão do problema que não permite ciclos degenerados, executamos os algoritmos da heurística e do exato para todas as combinações de cenário (ST e SL) e valores de $\alpha \in \{0{,}2; 0{,}4; 0{,}6; 0{,}8\}$. Os resultados são dados nas Tabelas de A.1 a A.8 do Apêndice.

Em cada uma das quatro tabelas com os resultados do cenário ST, conseguimos provar a otimalidade da solução de 7 das 28 instâncias dentro do tempo limite preestabelecido de 3600 segundos (uma hora). Com isso, o tempo médio de execução foi próximo ao tempo limite. Em todas essas 7 instâncias, a heurística foi capaz de encontrar o valor que se mostrou ótimo. Naturalmente, é possível que em algumas outras instâncias a heurística tenha encontrado uma solução ótima mas o algoritmo exato não tenha conseguido provar.

Há 3 instâncias em que o algoritmo exato conseguiu melhorar o valor encontrado pela heurística, todas no caso $\alpha = 0{,}2$. Em quatro oportunidades (três da *burma14* e uma da *gr17*) o algoritmo exato foi mais rápido que a heurística. Isso aconteceu porque as instâncias eram suficientemente pequenas para a solução da heurística ser ótima (e a heurística realizar muitas iterações desnecessárias) e as restrições do algoritmo exato conseguirem provar isso rapidamente. Em todas as outras instâncias, ocorreu o contrário, como esperado.

Nos casos ST com $\alpha = 0{,}6$ e $\alpha = 0{,}8$, a instância *bays29* se destacou com um número de cortes muito alto, o que é surpreendente, dado que não está entre as maiores.

Já o cenário SL obteve resultados melhores. Foi provado que encontramos uma solução ótima em 13 instâncias para $\alpha = 0{,}2$ e em 9 para cada outro caso. Em 9 dessas 13 oportunidades o algoritmo não precisou sair do nó raiz, ao contrário do que ocorreu em todas as instâncias do cenário ST. Em outras 35 instâncias, o algoritmo exato conseguiu melhorar o valor da solução. A maior redução foi de 6,7%, na instância *st70*, SL, $\alpha = 0{,}4$.

O valor do maior *gap* de integralidade foi de 20,21%, na instância *kroC100*, $\alpha = 0{,}8$.

4.3.2.2 Versão com ciclos degenerados

As Tabelas de A.9 a A.16 contêm os resultados obtidos na execução da heurística e do algoritmo exato para cada um dos cenários considerados, desta vez para a versão do problema que permite ciclos degenerados. Estes resultados foram quase sempre piores que os da outra versão, em critérios como número de instâncias resolvidas e valor do *gap*. Isso atesta a dificuldade maior desta versão.

No cenário ST, em todos os casos a heurística foi responsável pela melhor solução obtida. Para cada valor de α, três instâncias foram provadas ótimas. O *gap* de integralidade ficou muito alto, chegando a 42,05%. Foram de fato resolvidas 12 das 112 instâncias.

Já no cenário SL, o maior *gap* de integralidade caiu um pouco, para 36,34%, mas ainda

alto. O algoritmo exato conseguiu melhorar a solução da heurística em 10 ocasiões. Em quatro instâncias de cada tabela o exato conseguiu mostrar que a solução da heurística é ótima. Dezesseis instâncias foram totalmente resolvidas. A instância *burma14*, com $\alpha = 0{,}2$, precisou de apenas 1,47s no algoritmo exato, menos até do que os já breves 2,32s da heurística. Situação semelhante ocorreu com duas instâncias do caso $\alpha = 0{,}8$.

4.4 Comparações

Nesta seção, temos algumas comparações ilustrando como o comportamento e a estrutura da solução são afetados à medida que cada parâmetro varia. Nas Tabelas 4.2a, 4.2b e 4.2c as variações são, respectivamente, do parâmetro α, de cenário (ST ou SL) e de versão do problema (aceitando ou não ciclos degenerados).

Está claro que, de maneira geral, conforme α cresce, a dificuldade do problema aumenta: maior tempo de execução, maior número de cortes, maior número de nós explorados, maior *gap* de integralidade. De maneira geral, isso confirma o que é relatado na literatura, por exemplo no artigo publicado por de Camargo *et al.* (2013).

A diferença entre os cenários se dá na capacidade dos ciclos regulares: no ST, eles são bem apertados, enquanto que no SL há uma folga maior. Dessa forma, todas as soluções viáveis do ST também são viáveis no SL, mas o contrário não é verdade. Por isso, os valores das soluções ótimas do SL são sempre melhores (menores) que os correspondentes do cenário ST, como pode ser visto comparando as tabelas. O cenário SL costuma ser mais fácil de resolver, tendo usado um pouco menos de tempo, um *gap* de integralidade consideravelmente menor, número de nós menor e quantidade de cortes muito menor.

A tabela 4.2c comprova que o caso degenerado é, de fato, muito mais difícil. Fez uso de muito mais cortes, explorou muito mais nós, teve mais tempo de execução e, mesmo assim, seu *gap* de integralidade foi duas vezes pior. Como toda solução viável do caso não-degenerado é também viável para o caso degenerado, pode ser interessante usar o primeiro para resolver problemas para aplicações que aceitem o segundo, dependendo da necessidade de desempenho.

Nas Tabelas 4.2a, 4.2b e 4.2c, a melhor solução encontrada pela heurística possui valor praticamente igual ao da solução do algoritmo exato, executando em uma fração do tempo, o que atesta sua eficiência.

Tabela 4.2: Comparações de resultados variando cada um dos três aspectos.

(a) Comparações entre os resultados médios para cada valor de α.

α	Heurística		Branch-and-cut					Razão
	tempo (s)	valor	cortes	nós	tempo (s)	valor	*gap* (%)	
0,2	168,23	15138,72	45791,17	19296,77	2815,62	15027,17	14,72	1,00
0,4	177,85	16242,28	48047,66	19849,30	2872,96	16223,53	14,90	1,00
0,6	196,71	17348,33	51714,88	21523,62	2875,89	17311,10	15,37	1,00
0,8	201,89	18365,57	57406,49	22308,62	2876,15	18339,77	15,80	1,00

(b) Comparações entre os resultados médios para cada cenário (*ST* ou *SL*).

Cenário	Heurística		Branch-and-cut					Razão
	tempo (s)	valor	cortes	nós	tempo (s)	valor	*gap* (%)	
ST	117,60	17830,49	64068,47	23708,78	2975,51	17796,02	17,77	1,00
SL	254,74	15716,95	37411,62	17780,37	2744,80	15654,77	12,62	1,00

(c) Comparações entre os resultados médios para cada problema (sem ou com ciclos degenerados).

Problema	Heurística		Branch-and-cut					Razão
	tempo (s)	valor	cortes	nós	tempo (s)	valor	*gap* (%)	
Não Degenerado	206,92	17063,09	36178,77	16834,56	2545,83	16971,54	9,12	1,00
Degenerado	165,42	16484,36	65301,33	24654,59	3174,48	16479,25	21,28	1,00

Aumentar o valor de α faz o ciclo interligador tender a escolher arestas menores na solução ótima. Na Figura 4.3, em que a imagem inferior possui um valor de α maior, pode-se perceber essa característica visualmente.

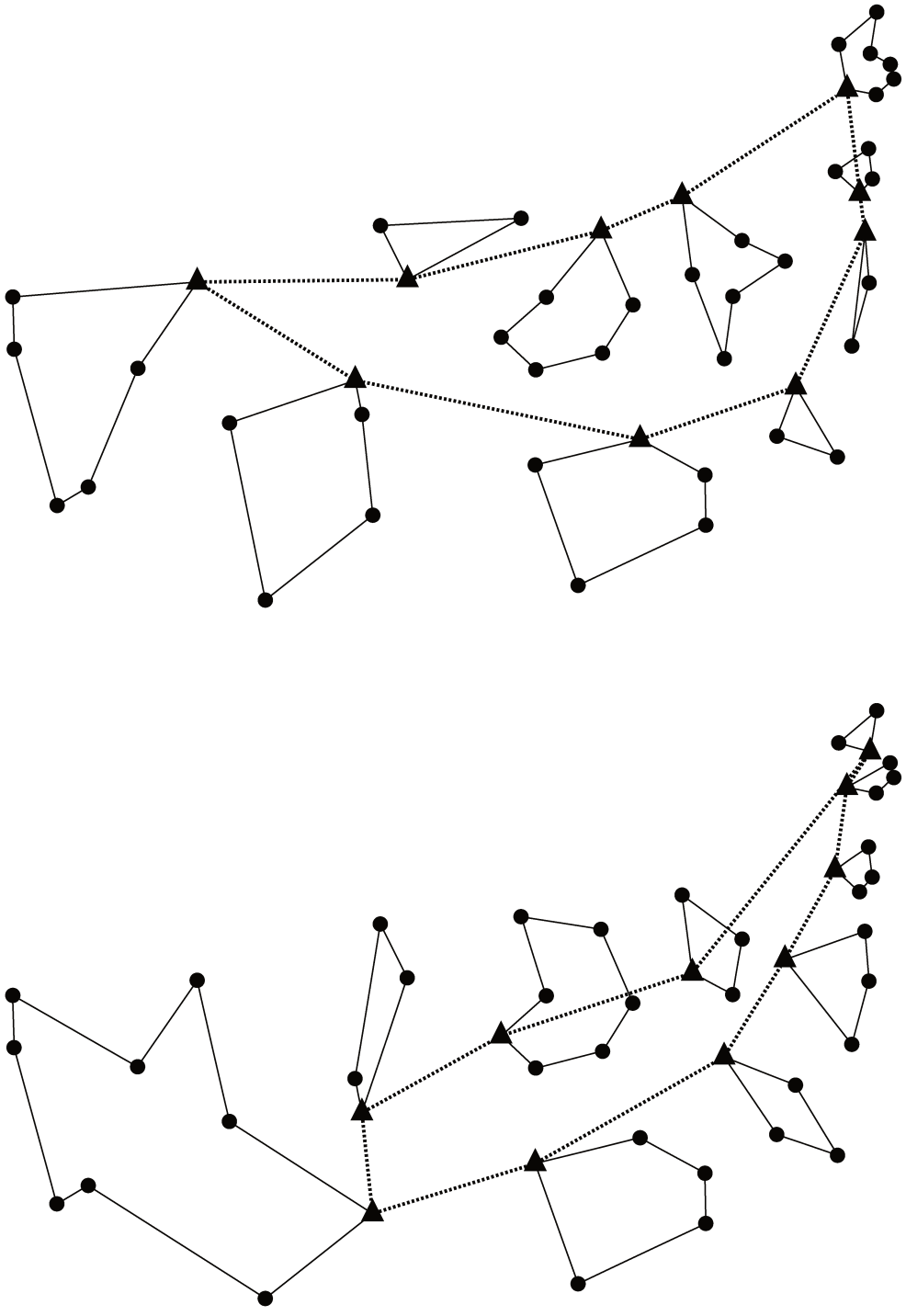


Figura 4.3: Comparação entre as melhores soluções viáveis obtidas para o grafo *att48* no cenário SL. Na figura superior, $\alpha = 0{,}2$, enquanto na inferior, $\alpha = 0{,}8$.

O cenário ST tende a ter ciclos regulares mais apertados que os do SL. Na Figura 4.4, note como a imagem superior (cenário ST) evita os ciclos grandes da inferior (SL).

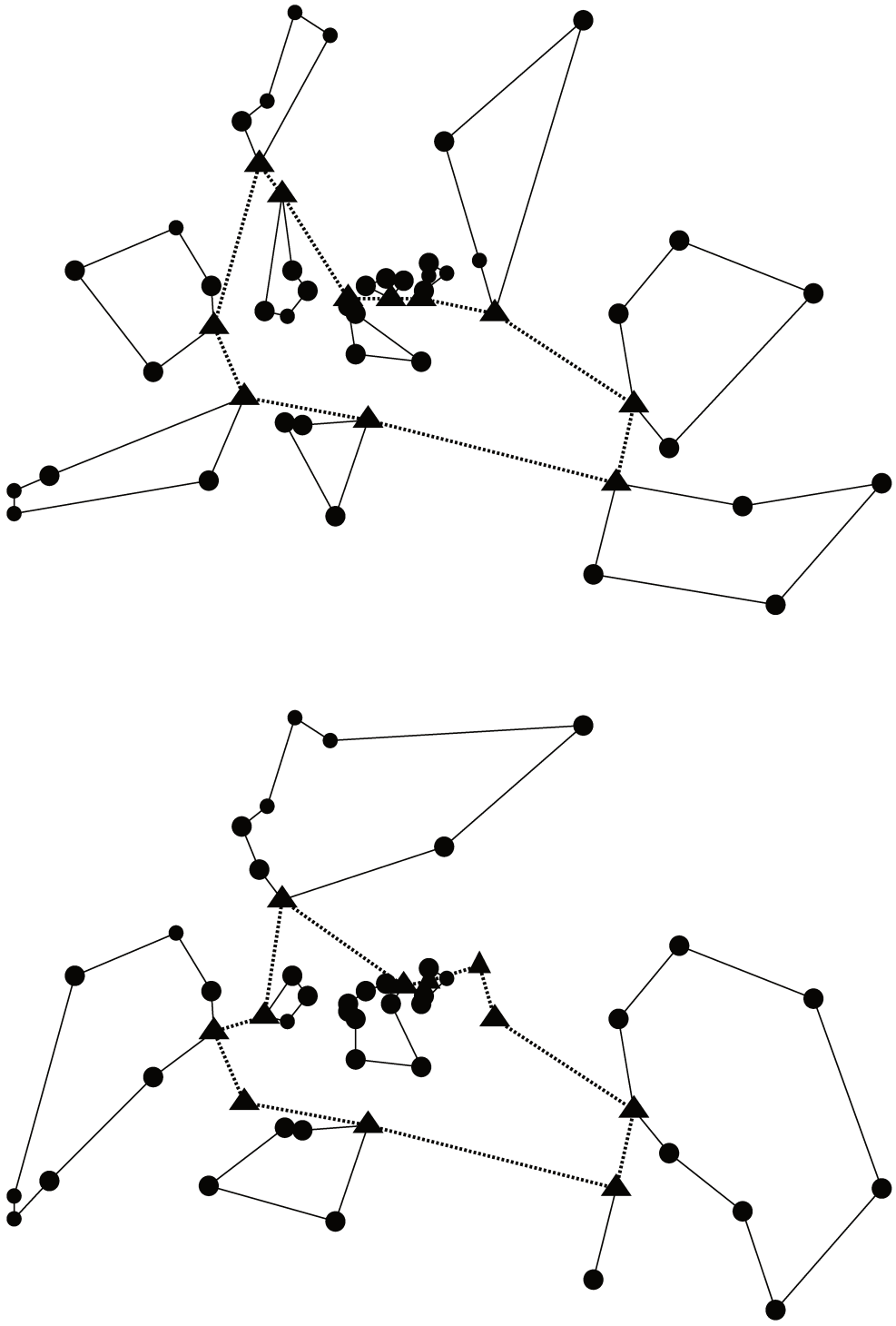


Figura 4.4: Melhores soluções viáveis obtidas para o grafo *berlin52*, com $\alpha = 0{,}8$. Os cenários são ST ($K = 11$, $C = 5$) e SL ($K = 11$, $C = 9$), respectivamente.

Por fim, a Figura 4.5 faz uma comparação mostrando como é grande a diferença da solução, para os mesmos parâmetros de entrada, apenas por permitir ou não ciclos degenerados. Somente a imagem inferior possui ciclos degenerados, cinco no total (dois com um cliente e três sem clientes).

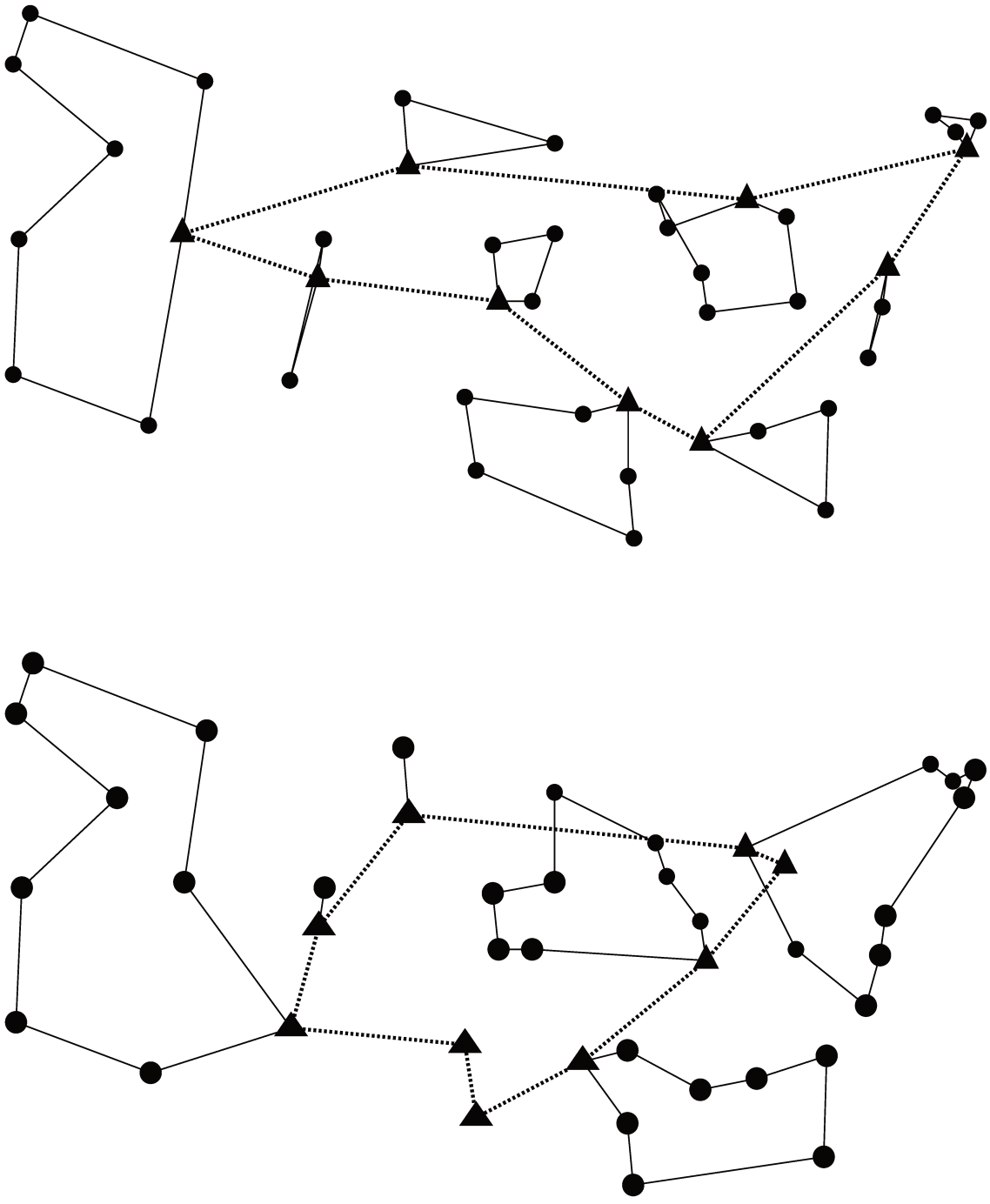


Figura 4.5: Melhores soluções viáveis encontradas para o grafo *dantzig42*, no cenário SL, com $\alpha = 0{,}4$, com ciclos degenerados apenas na segunda imagem.

Capítulo 5

Conclusão e Considerações Finais

Neste trabalho, analisamos diversos problemas fundamentais de roteamento de veículos e algumas técnicas comumente utilizadas para solucioná-los. Definimos duas variantes do nosso problema, com número de depósitos fixo ou variável, e escolhemos uma para estudar mais profundamente. Fizemos uma formulação em Programação Linear Inteira e mostramos que ela é correta. Também encontramos diversos cortes para fortalecer o modelo, mostrando as razões para serem válidos, bem como uma forma eficiente de separá-los.

Feito isso, implementamos uma heurística de busca local com solução inicial probabilística para encontrar boas soluções de maneira rápida. A busca local utilizou operações relativamente simples e bastante rápidas, como mover ou trocar vértices entre ciclos, além da tradicional heurística de Lin-Kernighan, esta bem mais sofisticada e pesada, originalmente utilizada no Problema do Caixeiro Viajante.

Implementamos também um algoritmo exato de *Branch-and-cut* para encontrar soluções ótimas do problema. Calibramos diversos aspectos do programa, desde a heurística até a ordem e critérios das rotinas de separação de cortes no algoritmo exato. Feito isso, executamos nosso programa com diversas instâncias-padrão da literatura e organizamos os resultados obtidos.

O algoritmo exato encontrou bastante dificuldade em resolver muitas das instâncias, mesmo estas sendo relativamente pequenas. Embora desanimador, isso não foi surpreendente, uma vez que outros artigos de referência, como o de Camargo *et al.* (2013), também não conseguiram resolver instâncias grandes e, além disso, nosso problema possui uma dificuldade adicional fundamental: o fato de todo vértice do grafo poder ser cliente ou depósito. Mesmo com essa dificuldade, conseguimos resolver na otimalidade 96 das 448 instâncias propostas (21,4%).

A surpresa positiva coube à heurística, a qual foi capaz de encontrar soluções viáveis muito boas e que em muitos casos vieram a se provar ótimas; quando isto não foi possível, raras

vezes o valor foi melhorado pelo algoritmo exato, o que pode ser visto como um indício de que o valor obtido pela heurística estava muito próximo (ou até igual) ao valor ótimo. Dessa forma, podemos recomendar o uso de nosso programa em casos reais de tamanho similar, desde que não seja imprescindível provar que a solução é ótima.

Trabalhos futuros podem tentar melhorar estes resultados ao aperfeiçoar os cortes, desenvolvendo novas desigualdades válidas ou rotinas de separação. É possível que alguma outra formulação permita uma execução mais rápida comparada ao algoritmo exato aqui proposto, como, por exemplo, a decomposição de Benders usada por de Camargo *et al.* (2013) ou algoritmos do tipo *branch-and-price*. Outra vertente para novos avanços consiste em implementar mais heurísticas, tais como VNS (*Variable Neighborhood Search*) e técnicas híbridas. Segundo Jarboui *et al.* (2013), eles foram os primeiros autores a propor uma heurística do tipo VNS para o problema integrado de localização e roteamento. Tal algoritmo permitiu obter soluções mais rapidamente que as heurísticas previamente consideradas para este problema, ao mesmo tempo que melhorou ou equiparou em termos de valor de solução.

Referências Bibliográficas

E. H. L. Aarts e J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. Discrete Mathematics and Optimization. Wiley, Chichester, UK, June 1997.

C. E. Andrade, F. K. Miyazawa e M. G. C. Resende. Evolutionary algorithm for the k-interconnected multi-depot multi-traveling salesmen problem. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, páginas 463–470, New York, NY, USA, 2013. ACM.

D. L. Applegate, W. Cook e A. Rohe. Chained lin-kernighan for large traveling salesman problems. *INFORMS J. on Computing*, 15(1):82–92, 2003.

D. L. Applegate, R. E. Bixby, V. Chvatal e W. J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

P. Augerat, A. Corberan, E. Benavent e J. M. Belenguer. Computational results with a branch and cut code for the capacitated vehicle routing problem. Relatório técnico RR 949-M, Université Grenoble 1. IMAG (Saint Martin d'Hères), 1995.

P. Bauer, J. T. Linderoth e M. W. P. Savelsbergh. A branch and cut approach to the cardinality constrained circuit problem. *Mathematical programming*, 91(2):307–348, 2002.

J. M. Belenguer, E. Benavent, C. Prins, C. Prodhon e R. W. Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, 38:931–941, 2011.

E. Benavent e A. Martínez. Multi-depot Multiple TSP: a polyhedral study and computational results. *Annals of Operations Research*, páginas 1–19, Novembro 2011.

A. Bruns, A. Klose e P. Stähly. Restructuring of swiss parcel delivery services. *OR Spektrum*, 22:285–302, 2000.

S. Çetiner, C. Sepil e H. Süral. Hubbing and postal routing in postal delivery systems. *Annals of Operations Research*, 181:109–124, 2010.

Y. Chan e S. F. Baker. The multiple depot, multiple traveling salesmen facility-location problem: Vehicle range, service frequency, and heuristic implementations. *Math. Comput. Model.*, 41(8-9):1035–1053, Abril 2005.

C. Contardo, V. Hemmelmayr e T. G. Crainic. Lower and upper bounds for the two-echelon capacitated location-routing problem. *Computers & Operations Research*, 39:3185–3199, 2012.

C. Contardo, J. F. Cordeaub e B. Gendron. A computational comparison of flow formulations for the capacitated location-routing problem. *Discrete Optimization*, 10(4):263–295, 2013.

I. Contreras, E. Fernández e A. Marín. The tree of hubs location problem. *European Journal of Operational Research*, 202(2):390 – 400, 2010. ISSN 0377-2217.

W. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press. Princeton University Press, 2011.

G. B. Dantzig e J. H. Ramser. The truck dispatching problem. *Management Science*, 6: 80–91, 10 1959.

S. Dasgupta, C. H. Papadimitriou e U. V. Vazirani. *Algorithms*. McGraw-Hill, 2008. ISBN 978-0-07-352340-8.

R. S. de Camargo, G. de Miranda e A. Løkketangen. A new formulation and an exact approach for the many-to-many hub location-routing problem. *Applied Mathematical Modelling*, 37(12-13):7465 – 7480, 2013.

B. Dezső, A. Jüttner e P. Kovács. LEMON - an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23 – 45, 2011.

A. G. Doig, B. H. Land e A. G. Doig. An automatic method for solving discrete programming problems. *Econometrica*, páginas 497–520, 1960.

A. S. Fraser. Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian Journal of Biological Science*, 10:484–491, 1957.

M. R. Garey e D. S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. San Francisco: Freeman, 1979.

R. E. Gomory e T. C. Hu. Multi-Terminal Network Flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570, 1961.

J. F. Gonçalves e M. G. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, Outubro 2011.

Gurobi. Gurobi optimizer reference manual, 2013. URL `http://www.gurobi.com`.

K. Helsgaun. An effective implementation of the lin-kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126:106–130, 2000.

IBM. *ILOG® CPLEX® V12.1 - User's Manual for CPLEX®*. IBM Corporation, 2009.

B. Jarboui, H. Derbel, S. Hanafi e N. Mladenović. Variable neighborhood search for location routing. *Computers & Operations Research*, 40(1):47 – 57, 2013. ISSN 0305-0548.

I. Karaoglan, F. Altiparmak, I. Kara e B. Dengiz. The location-routing problem with simultaneous pickup, delivery: formulations, a heuristic approach. *Omega*, 40(4):465–477, 2012.

M. J. Kuby e R. G. Gray. The hub network design problem with stopovers and feeders: the case of federal express. *TOP*, 27:1–12, 1993.

E. Lawler. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley, New York, 1985.

C. C. Lin, J. Y. Lin e Y. C. Chen. The capacitated p-hub median problem with integral constraints: an application to a chinese air cargo network. *Applied Mathematical Modelling*, 36:2777–2787, 2012.

R. B. Lopes, C. Ferreira, B. S. Santos e S. Barreto. A taxonomical analysis, current methods and objectives on location-routing problems. *International Transactions in Operational Research*, 20:795–822, 2013.

R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM J. Res. Dev.*, 47(1):57–66, Janeiro 2003.

J. Lysgaard, A. N. Letchford e R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

Y. Z. Mehrjerdi e A. Nadizadeh. Using greedy clustering method to solve capacitated location-routing problem with fuzzy demands. *European Journal of Operational Research*, 229: 75–84, 2013.

H. Min, V. Jayaramanb e R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108:1–15, 1998.

J. E. Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of Applied Optimization*, páginas 65–77, 2002.

D. Naddef e G. Rinaldi. The vehicle routing problem. capítulo Branch-and-cut Algorithms for the Capacitated VRP, páginas 53–84. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

G. Nagy e S. Salhi. Location-routing: Issues, models and methods. *European Journal of Operational Research*, 177:649–672, 2007.

G. Nagy e S. Salhi. The many-to-many location-routing problem. *TOP*, 6:261–275, 1998.

V. P. Nguyen, C. Prins e C. Prodhon. Solving the two-echelon location routing problem by a grasp reinforced by a learning process and path relinking. *European Journal of Operational Research*, 216:113–126, 2012.

M. E. O'Kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32:393–404, 1987.

G. Reinelt. TSPLIB - A Traveling Salesman Problem Library. *INFORMS Journal on Computing*, 3(4):376–384, 1991.

I. Rodríguez-Martín, J. J. Salazar-González e H. Yaman. A branch-and-cut algorithm for the hub location and routing problem. *Computers & Operations Research*, 50(0):161 – 174, 2014. ISSN 0305-0548.

F. Rothlauf. *Design of Modern Heuristics*. Natural Computing Series. Springer, 2011. ISBN 978-3-540-72961-7.

A. G. Thomason. Hamiltonian cycles and uniquely edge colourable graphs. In B. Bollobás, editor, *Advances in Graph Theory*, volume 3 of *Annals of Discrete Mathematics*, páginas 259 – 268. Elsevier, 1978.

Z. Wang, C. Lin e C. K. Chan. Demonstration of a single-fiber self-healing cwdm metro access ring network with unidirectional oadm. *Photonics Technology Letters*, 18:163–165, 2006.

M. Wasner e G. Zaäpfel. An integrated multi-depot hub-location vehicle routing model for network planning of parcel service. *International Journal of Production Economics*, 90: 403–419, 2004.

V. F. Yu, S. W. Lin, W. Lee e C. J. Ting. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, 58:288–299, 2010.

M. H. F. Zarandi, A. Hemmati, S. Davari e I. B. Turksen. Capacitated location-routing problem with time windows under uncertainty. *Knowledge-Based Systems*, 37:480–489, 2013.

Apêndice A

Tabelas de resultados

Tabela A.1: Resultados para o cenário ST com $\alpha = 0{,}2$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	143,24	12260	22931	25142	3600,40	12250,2	7,74	1,00
bayg29	29	6	5	16,41	1807,4	17430	51248	3600,08	1807,4	3,98	1,00
bays29	29	6	5	10,96	2293,8	17732	52303	3600,09	2293,8	3,85	1,00
berlin52	52	11	5	105,72	8929,2	24816	18730	3600,53	8929,2	10,92	1,00
brazil58	58	12	5	164,86	26611,4	28520	20982	3600,70	26611,4	8,18	1,00
burma14	14	3	5	2,95	$3680{,}2^{+}$	784	1686	0,88	3680,2*	0,00	1,00
dantzig42	42	9	5	32,16	814,4	23136	25659	3600,06	807	9,47	1,01
eil51	51	11	5	80,05	486,8	25351	21115	3600,57	486,8	6,29	1,00
eil76	76	16	5	154,70	647	32863	12396	3601,49	647	11,22	1,00
fri26	26	6	5	9,96	$1016{,}4^{+}$	4211	9961	131,44	1016,4*	0,00	1,00
gr17	17	4	5	5,12	$1997{,}2^{+}$	263	188	0,38	1997,2*	0,00	1,00
gr21	21	5	5	7,06	$3182{,}4^{+}$	1428	3230	18,08	3182,4*	0,00	1,00
gr24	24	5	5	6,73	$1469{,}6^{+}$	5092	11277	117,11	1469,6*	0,00	1,00
gr48	48	10	5	43,30	5671,2	26491	25407	3600,85	5667,6	7,51	1,00
gr96	96	20	5	377,53	68425	51099	9505	3602,36	68425	16,71	1,00
hk48	48	10	5	64,74	13492,2	27823	25969	3600,35	13492,2	6,97	1,00
kroA100	100	20	5	466,42	28397,6	31852	5424	3606,28	28397,6	21,41	1,00
kroB100	100	20	5	255,40	28687,4	27224	6928	3603,39	28687,4	19,90	1,00
kroC100	100	20	5	151,37	28428	24686	3256	3600,24	28428	23,54	1,00
kroD100	100	20	5	153,43	29191,8	46913	3421	3600,27	29191,8	23,62	1,00
kroE100	100	20	5	286,01	29682	51869	8362	3603,13	29682	22,04	1,00
pr76	76	16	5	147,24	130884	41802	14029	3600,87	123510	10,42	1,06
rat99	99	20	5	163,92	1625,4	28083	6914	3602,72	1623	19,16	1,00
rd100	100	20	5	309,77	10312,8	67405	8326	3600,10	10312,8	19,83	1,00
st70	70	14	5	50,47	837,8	34601	12548	3600,79	837,8	15,56	1,00
swiss42	42	9	5	27,65	1491,6	21862	27640	3600,27	1478,8	7,41	1,01
ulysses16	16	4	4	3,79	$7766{,}2^{+}$	1929	7671	39,06	7766,2*	0,00	1,00
ulysses22	22	5	5	8,41	$7165{,}6^{+}$	3339	6850	57,93	7165,6*	0,00	1,00
Média	–	–	–	**116,05**	–	**24697,68**	**15220,25**	**2713,94**	–	**9,85**	**1,00**

Tabela A.2: Resultados para o cenário ST com $\alpha = 0{,}4$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	80,54	13271,4	39813	29337	3600,42	13271,4	12,18	1,00
bayg29	29	6	5	9,64	1916	120033	111417	3600,08	1916	2,01	1,00
bays29	29	6	5	10,53	2429,8	71579	76977	3600,06	2429,8	5,60	1,00
berlin52	52	11	5	192,73	9417,2	33875	21969	3600,53	9416,2	10,23	1,00
brazil58	58	12	5	124,62	29064,6	66986	19718	3600,40	29064,6	10,34	1,00
burma14	14	3	5	2,93	$3891{,}4^{+}$	1816	1834	3,09	3891,4*	0,00	1,00
dantzig42	42	9	5	41,42	881,6	42976	28657	3600,25	881,6	10,89	1,00
eil51	51	11	5	73,88	522,4	27592	23028	3600,39	522,4	9,11	1,00
eil76	76	16	5	154,73	683,6	42217	15989	3601,52	683,6	13,08	1,00
fri26	26	6	5	10,19	$1086{,}8^{+}$	6708	23839	210,25	1086,8*	0,00	1,00
gr17	17	4	5	5,17	$2139{,}8^{+}$	471	703	0,57	2139,8*	0,00	1,00
gr21	21	5	5	7,13	$3396{,}2^{+}$	5125	8073	50,11	3396,2*	0,00	1,00
gr24	24	5	5	6,87	$1542{,}2^{+}$	11546	26390	267,30	1542,2*	0,00	1,00
gr48	48	10	5	101,62	6031,2	50042	30169	3600,56	6031,2	9,25	1,00
gr96	96	20	5	379,65	71030,2	37763	8936	3601,71	71030,2	16,69	1,00
hk48	48	10	5	120,85	14358,4	27903	26936	3600,46	14358,4	9,40	1,00
kroA100	100	20	5	186,40	30728	29239	2983	3608,54	30728	24,30	1,00
kroB100	100	20	5	166,51	32223,6	31688	2336	3600,57	32223,6	26,05	1,00
kroC100	100	20	5	197,43	30699,6	36876	3210	3600,32	30699,6	26,40	1,00
kroD100	100	20	5	434,39	30103,2	24431	3912	3604,28	30103,2	23,56	1,00
kroE100	100	20	5	257,62	31677,2	123179	10839	3601,39	31677,2	24,17	1,00
pr76	76	16	5	174,25	136679	43778	14304	3601,05	136679	14,54	1,00
rat99	99	20	5	255,72	1664	31675	4829	3600,44	1664	18,85	1,00
rd100	100	20	5	194,82	11378	59851	7851	3600,76	11378	25,18	1,00
st70	70	14	5	76,87	895,4	136595	19514	3600,89	893,4	18,96	1,00
swiss42	42	9	5	35,69	1568,8	40191	33834	3600,29	1568,8	8,67	1,00
ulysses16	16	4	4	3,77	$8073{,}4^{+}$	4063	5126	13,10	8073,4*	0,00	1,00
ulysses22	22	5	5	7,53	$7544{,}2^{+}$	5356	10885	84,67	7544,2*	0,00	1,00
Média	–	–	–	**118,34**	–	**41191,68**	**20485,54**	**2723,36**	–	**11,41**	**1,00**

Tabela A.3: Resultados para o cenário ST com $\alpha = 0{,}6$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	74,05	14219,4	68074	36508	3600,10	14219,4	14,08	1,00
bayg29	29	6	5	10,07	2014	135001	106673	3600,06	2014	4,10	1,00
bays29	29	6	5	8,77	2545,2	143564	101217	3600,20	2545,2	4,48	1,00
berlin52	52	11	5	68,56	10003,8	51485	25440	3600,43	10003,8	11,01	1,00
brazil58	58	12	5	130,59	31379,8	65679	19932	3600,45	31232,4	11,80	1,00
burma14	14	3	5	2,94	$4102{,}6^{+}$	2579	2342	0,78	4102,6*	0,00	1,00
dantzig42	42	9	5	24,29	943,4	40760	30765	3600,38	943,4	13,25	1,00
eil51	51	11	5	138,43	543,2	32849	25598	3600,13	543,2	9,79	1,00
eil76	76	16	5	115,04	715,8	40544	13719	3601,24	714	13,75	1,00
fri26	26	6	5	10,05	$1157{,}2^{+}$	24899	47343	462,30	1157,2*	0,00	1,00
gr17	17	4	5	5,12	$2270{,}2^{+}$	663	692	0,62	2270,2*	0,00	1,00
gr21	21	5	5	7,14	$3590{,}8^{+}$	8010	7728	34,20	3590,8*	0,00	1,00
gr24	24	5	5	7,05	$1614{,}8^{+}$	41039	48677	435,95	1614,8*	0,00	1,00
gr48	48	10	5	177,46	6401,6	79237	35784	3600,37	6401,6	10,68	1,00
gr96	96	20	5	276,41	78833,6	33182	8298	3600,73	78711,6	22,73	1,00
hk48	48	10	5	122,66	15224,6	57002	38063	3600,29	15224,6	9,48	1,00
kroA100	100	20	5	197,28	33208,6	35452	3929	3602,06	33208,6	27,85	1,00
kroB100	100	20	5	418,69	33579,6	53680	4248	3601,76	33579,6	26,48	1,00
kroC100	100	20	5	264,47	32994,8	45709	3770	3600,08	32994,8	29,04	1,00
kroD100	100	20	5	269,71	32654	34228	6437	3602,33	32654	26,80	1,00
kroE100	100	20	5	175,11	33717,6	32646	3784	3604,98	33717,6	26,53	1,00
pr76	76	16	5	139,93	145048	46541	14499	3600,74	145048	15,73	1,00
rat99	99	20	5	179,05	1798	24459	2640	3600,12	1798	21,55	1,00
rd100	100	20	5	176,22	11866,4	28077	2587	3600,32	11866,4	25,45	1,00
st70	70	14	5	88,50	913,8	90323	18662	3600,10	913,8	17,01	1,00
swiss42	42	9	5	42,74	1678	109399	38475	3600,20	1670,4	10,18	1,00
ulysses16	16	4	4	4,64	$8344{,}6^{+}$	5164	5527	8,93	8344,6*	0,00	1,00
ulysses22	22	5	5	7,52	$7849{,}8^{+}$	6262	8517	44,34	7849,8*	0,00	1,00
Média	–	–	–	**112,23**	–	**47732,39**	**23637,64**	**2735,86**	–	**12,56**	**1,00**

Tabela A.4: Resultados para o cenário ST com $\alpha = 0{,}8$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	65,09	15142,6	108922	33580	3600,42	15142,6	14,87	1,00
bayg29	29	6	5	10,14	2112	179586	86501	3600,23	2112	3,83	1,00
bays29	29	6	5	8,92	2654,4	284784	109033	3600,01	2654,4	4,22	1,00
berlin52	52	11	5	162,82	10341,8	61015	25754	3600,44	10341,8	9,98	1,00
brazil58	58	12	5	262,40	33409,8	60086	25068	3600,57	33387,8	11,45	1,00
burma14	14	3	5	2,96	4270,8^{+}	3397	3494	2,80	4270,8*	0,00	1,00
dantzig42	42	9	5	29,70	1007,4	87491	33611	3600,06	1000,8	11,89	1,01
eil51	51	11	5	81,64	570,8	47372	28191	3600,66	570,8	10,79	1,00
eil76	76	16	5	84,73	763	134983	18192	3600,77	763	16,44	1,00
fri26	26	6	5	10,37	1227,6^{+}	75988	51044	437,73	1227,6*	0,00	1,00
gr17	17	4	5	5,24	2392,6^{+}	603	675	0,66	2392,6*	0,00	1,00
gr21	21	5	5	7,18	3780^{+}	17262	12785	59,46	3780*	0,00	1,00
gr24	24	5	5	8,25	1676,6^{+}	62483	55642	492,75	1676,6*	0,00	1,00
gr48	48	10	5	50,51	6773,4	55133	32468	3600,45	6773,4	13,44	1,00
gr96	96	20	5	216,63	83077,2	36778	6038	3605,57	83077,2	24,50	1,00
hk48	48	10	5	59,56	16090,8	69367	33375	3600,38	16090,8	11,20	1,00
kroA100	100	20	5	326,78	34903,8	32424	3980	3600,40	34903,8	29,58	1,00
kroB100	100	20	5	318,68	35018,6	74196	6603	3600,25	35018,6	27,36	1,00
kroC100	100	20	5	291,28	34484,2	28555	3296	3600,83	34484,2	29,60	1,00
kroD100	100	20	5	287,47	34550,8	25073	2616	3605,71	34550,8	28,75	1,00
kroE100	100	20	5	230,54	34943,2	143500	11676	3602,24	34943,2	26,88	1,00
pr76	76	16	5	249,31	154459	49496	14617	3600,34	154459	17,83	1,00
rat99	99	20	5	488,90	1855	13849	4486	3600,46	1855	21,57	1,00
rd100	100	20	5	242,52	12229,6	23475	3438	3601,01	12229,6	26,54	1,00
st70	70	14	5	49,84	963,2	66428	20654	3600,32	963,2	18,90	1,00
swiss42	42	9	5	41,95	1753,6	86727	38179	3600,36	1753,6	10,86	1,00
ulysses16	16	4	4	3,74	8615,8^{+}	4020	4685	8,66	8615,8*	0,00	1,00
ulysses22	22	5	5	7,61	8155,4^{+}	7809	8001	32,12	8155,4*	0,00	1,00
Média	–	–	–	**128,74**	–	**65742,93**	**24202,93**	**2737,70**	–	**13,23**	**1,00**

Tabela A.5: Resultados para o cenário SL com $\alpha = 0{,}2$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	164,50	11260,2	23265	26842	3600,43	11189	2,51	1,01
bayg29	29	6	9	22,03	1703,6$^+$	7213	10987	170,41	1703,6*	0,00	1,00
bays29	29	6	9	17,96	2113,2$^+$	2588	3442	36,23	2113,2*	0,00	1,00
berlin52	52	11	9	182,48	8033,2	6151	8189	611,83	7919,8*	0,00	1,01
brazil58	58	12	9	342,99	24535,8	6863	7816	655,88	23755*	0,00	1,03
burma14	14	3	9	3,04	3145,2$^+$	6	0	0,01	3145,2*	0,00	1,00
dantzig42	42	9	9	60,23	719,4$^+$	4511	6571	228,70	719,4*	0,00	1,00
eil51	51	11	9	132,62	472,2	30680	21244	3601,02	471	5,14	1,00
eil76	76	16	9	446,54	606,6	20770	13831	3600,75	603,8	6,36	1,00
fri26	26	6	9	12,56	973,4	656	903	2,22	973*	0,00	1,00
gr17	17	4	9	5,41	1887,6$^+$	37	0	0,07	1887,6*	0,00	1,00
gr21	21	5	9	7,59	2987$^+$	603	678	1,24	2987*	0,00	1,00
gr24	24	5	9	10,14	1314,2$^+$	115	114	0,49	1314,2*	0,00	1,00
gr48	48	10	9	133,19	5435,8	31610	24775	3600,75	5425,2	5,12	1,00
gr96	96	20	9	786,78	64382,4	22043	7814	3605,15	64382,4	13,09	1,00
hk48	48	10	9	132,22	12575	18978	24752	3600,25	12366,8	1,71	1,02
kroA100	100	20	9	870,97	23841,6	14206	7749	3601,13	23841,6	10,48	1,00
kroB100	100	20	9	514,33	25230,2	18248	8696	3604,11	24946,8	11,44	1,01
kroC100	100	20	9	651,30	24641,6	20237	7739	3603,46	24395,8	12,81	1,01
kroD100	100	20	9	898,62	24721,8	18487	8028	3604,96	24721,8	12,79	1,00
kroE100	100	20	9	575,55	25584,8	18850	7328	3602,05	25282,6	12,34	1,01
pr76	76	16	9	649,91	119494	25634	12996	3601,85	117038	6,59	1,02
rat99	99	20	9	464,85	1417,2	16429	7797	3600,41	1417,2	9,86	1,00
rd100	100	20	9	507,42	9045,8	17789	7454	3601,23	8646,4	7,86	1,05
st70	70	14	9	140,50	737,8	22192	14784	3600,77	696,6	3,39	1,06
swiss42	42	9	9	72,98	1371,8	12370	30377	3190,33	1360*	0,00	1,01
ulysses16	16	4	7	4,66	6467$^+$	27	0	0,03	6467*	0,00	1,00
ulysses22	22	5	9	10,23	6598,4$^+$	64	0	0,24	6598,4*	0,00	1,00
Média	–	–	–	**279,34**	–	**12879,36**	**9675,21**	**2104,50**	–	**4,34**	**1,01**

Tabela A.6: Resultados para o cenário SL com $\alpha = 0{,}4$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	132,25	12187,2	53165	26880	3600,85	12152,2	6,44	1,00
bayg29	29	6	9	15,64	1799,2$^+$	8774	9351	115,44	1799,2*	0,00	1,00
bays29	29	6	9	16,13	2238,4$^+$	7266	7545	81,53	2238,4*	0,00	1,00
berlin52	52	11	9	299,81	8603,2	20439	22485	3603,76	8584,6	3,97	1,00
brazil58	58	12	9	457,48	26403,6	29895	41097	3600,31	26231	0,34	1,01
burma14	14	3	9	3,02	3289,4$^+$	7	0	0,01	3289,4*	0,00	1,00
dantzig42	42	9	9	58,24	802,6	23805	34370	3600,12	790,4	3,29	1,02
eil51	51	11	9	100,05	501	39376	25542	3600,81	501	6,87	1,00
eil76	76	16	9	236,38	641,8	44149	17299	3600,53	630,6	6,79	1,02
fri26	26	6	9	11,91	1036,6$^+$	2082	2864	22,61	1036,6*	0,00	1,00
gr17	17	4	9	5,29	2048,8$^+$	49	0	0,19	2048,8*	0,00	1,00
gr21	21	5	9	7,52	3142$^+$	912	1591	3,95	3142*	0,00	1,00
gr24	24	5	9	10,83	1391,4$^+$	282	216	0,50	1391,4*	0,00	1,00
gr48	48	10	9	186,53	5785,4	36738	29582	3600,33	5723	5,69	1,01
gr96	96	20	9	422,57	67794	13784	6710	3603,42	67794	14,25	1,00
hk48	48	10	9	105,55	13289,6	51330	28086	3600,28	13255	4,81	1,00
kroA100	100	20	9	901,09	25882,2	13473	5592	3600,98	25039,2	10,63	1,03
kroB100	100	20	9	814,57	26411,4	19118	7791	3601,43	26111,4	11,10	1,01
kroC100	100	20	9	638,49	26298,8	22191	7653	3602,27	25986,2	14,82	1,01
kroD100	100	20	9	490,61	26618,4	24799	7555	3603,50	26618,4	15,37	1,00
kroE100	100	20	9	1029,62	27096	38105	8059	3601,34	27096	14,44	1,00
pr76	76	16	9	276,70	127495	36484	14198	3600,50	127495	9,59	1,00
rat99	99	20	9	572,97	1504,6	10544	5702	3605,51	1504,6	11,66	1,00
rd100	100	20	9	773,37	9494,8	32479	8567	3602,83	9494,8	12,35	1,00
st70	70	14	9	170,92	792	22750	13107	3600,31	738,8	4,17	1,07
swiss42	42	9	9	43,01	1487,8	49509	35946	3600,29	1466,6	3,61	1,01
ulysses16	16	4	7	4,69	6821$^+$	31	0	0,09	6821*	0,00	1,00
ulysses22	22	5	9	9,72	6991,8$^+$	231	225	0,48	6991,8*	0,00	1,00
Média	–	–	–	**278,39**	–	**21491,68**	**13143,32**	**2451,93**	–	**5,72**	**1,01**

Tabela A.7: Resultados para o cenário SL com $\alpha = 0{,}6$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	122,67	13016,6	71577	30858	3600,38	13016,6	7,53	1,00
bayg29	29	6	9	18,45	1875,8$^+$	5198	4408	35,93	1875,8*	0,00	1,00
bays29	29	6	9	14,58	2353,8$^+$	8266	6429	53,80	2353,8*	0,00	1,00
berlin52	52	11	9	259,96	9219,8	37715	25653	3601,39	9159,6	6,11	1,01
brazil58	58	12	9	639,61	29469,4	43829	24174	3600,65	28728,6	5,92	1,03
burma14	14	3	9	3,03	3433,6$^+$	7	0	0,01	3433,6*	0,00	1,00
dantzig42	42	9	9	39,07	854,2	43226	34110	3600,31	854,2	6,95	1,00
eil51	51	11	9	100,49	525,2	46910	28456	3600,52	525,2	7,16	1,00
eil76	76	16	9	192,49	678	48241	16368	3600,15	670	8,81	1,01
fri26	26	6	9	11,78	1092,4$^+$	4942	5223	35,96	1092,4*	0,00	1,00
gr17	17	4	9	5,23	2167,2$^+$	65	29	0,18	2167,2*	0,00	1,00
gr21	21	5	9	7,53	3297$^+$	1841	1910	11,07	3297*	0,00	1,00
gr24	24	5	9	13,13	1468,6$^+$	1289	1453	2,92	1468,6*	0,00	1,00
gr48	48	10	9	196,71	6028,2	81836	35847	3600,35	6016,2	5,99	1,00
gr96	96	20	9	813,53	70611	32610	6355	3600,12	70611	15,13	1,00
hk48	48	10	9	269,87	13950,6	41907	32589	3600,82	13950,6	4,65	1,00
kroA100	100	20	9	884,37	27415,6	36699	8556	3602,62	27334,6	14,43	1,00
kroB100	100	20	9	947,39	28658,8	18639	6390	3602,86	28658,8	15,73	1,00
kroC100	100	20	9	577,65	28074,8	20790	6187	3605,81	27859,8	17,42	1,01
kroD100	100	20	9	656,05	28123,4	32994	7600	3603,45	27529,4	14,73	1,02
kroE100	100	20	9	767,63	28646	27256	5051	3609,02	28646	15,76	1,00
pr76	76	16	9	387,68	138619	35708	15569	3602,27	136814	12,44	1,01
rat99	99	20	9	875,88	1587	18708	6991	3602,34	1584	12,73	1,00
rd100	100	20	9	368,72	10238,2	26164	6579	3600,65	10200,2	15,28	1,00
st70	70	14	9	335,64	824	44429	18010	3601,43	820,8	10,09	1,00
swiss42	42	9	9	33,29	1580	133200	46369	3600,22	1569,8	5,76	1,01
ulysses16	16	4	7	4,73	7175$^+$	84	69	0,17	7175*	0,00	1,00
ulysses22	22	5	9	9,87	7385,2$^+$	535	606	0,96	7385,2*	0,00	1,00
Média	–	–	–	**305,61**	–	**30880,89**	**13637,11**	**2449,16**	–	**7,24**	**1,00**

Tabela A.8: Resultados para o cenário SL com $\alpha = 0{,}8$, sem ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	269,24	13889	76756	32656	3600,41	13827,8	10,70	1,00
bayg29	29	6	9	17,29	1952,4$^+$	4315	3664	28,80	1952,4*	0,00	1,00
bays29	29	6	9	14,77	2440,4$^+$	6130	4931	39,18	2440,4*	0,00	1,00
berlin52	52	11	9	259,70	9678,2	141571	33241	3600,48	9596,4	8,00	1,01
brazil58	58	12	9	303,21	31475	108703	25983	3600,35	31210,6	11,23	1,01
burma14	14	3	9	3,09	3577,8$^+$	11	0	0,01	3577,8*	0,00	1,00
dantzig42	42	9	9	49,73	912,4	86255	39627	3600,32	908,4	6,58	1,00
eil51	51	11	9	110,14	551,4	68735	30222	3600,46	551,4	8,16	1,00
eil76	76	16	9	170,01	708,6	54813	16894	3600,90	703,6	10,35	1,01
fri26	26	6	9	11,44	1148,2$^+$	9824	8218	64,55	1148,2*	0,00	1,00
gr17	17	4	9	5,31	2285,6$^+$	287	217	0,20	2285,6*	0,00	1,00
gr21	21	5	9	7,87	3452$^+$	1631	1497	3,11	3452*	0,00	1,00
gr24	24	5	9	11,55	1545,8$^+$	4125	4229	23,78	1545,8*	0,00	1,00
gr48	48	10	9	156,42	6430	128809	37113	3600,43	6424	7,48	1,00
gr96	96	20	9	435,96	75505,4	56740	9496	3602,85	75286,4	17,95	1,00
hk48	48	10	9	173,78	14714	57597	35738	3600,08	14695	5,71	1,00
kroA100	100	20	9	870,69	29866,4	22281	6281	3615,48	29866,4	19,01	1,00
kroB100	100	20	9	1000,65	29943,8	13270	6564	3601,59	29678,8	15,49	1,01
kroC100	100	20	9	885,77	29768,6	20113	6393	3601,06	29716,6	20,21	1,00
kroD100	100	20	9	1097,22	30192,2	43502	7971	3600,75	29521,2	18,24	1,02
kroE100	100	20	9	576,79	30861,2	20013	3270	3600,77	30861,2	19,11	1,00
pr76	76	16	9	236,46	144965	68711	16522	3600,57	144230	13,56	1,01
rat99	99	20	9	976,52	1685,8	24320	7343	3606,38	1685,8	15,02	1,00
rd100	100	20	9	874,64	10690,4	56148	8842	3601,35	10690,4	16,99	1,00
st70	70	14	9	296,51	870,2	54444	17382	3600,63	863,8	10,81	1,01
swiss42	42	9	9	37,97	1657,4	123390	43936	3600,30	1636,8	5,84	1,01
ulysses16	16	4	7	4,68	7484,4$^+$	188	182	0,18	7484,4*	0,00	1,00
ulysses22	22	5	9	9,61	7778,6$^+$	2097	2474	10,96	7778,6*	0,00	1,00
Média	–	–	–	**316,68**	–	**44813,54**	**14674,50**	**2450,21**	–	**8,59**	**1,00**

Tabela A.9: Resultados para o cenário ST com $\alpha = 0{,}2$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	78,52	12250,4	119898	24263	3601,72	12250,4	26,16	1,00
bayg29	29	6	5	8,76	1807,4	190901	61891	3600,06	1807,4	10,10	1,00
bays29	29	6	5	8,19	2293,8	161956	60511	3600,09	2293,8	10,68	1,00
berlin52	52	11	5	88,17	8573,4	98077	16376	3600,31	8573,4	24,80	1,00
brazil58	58	12	5	139,90	26224,8	98405	15608	3601,03	26223,4	24,33	1,00
burma14	14	3	5	2,89	3680,2^{+}	10904	9515	23,28	3680,2*	0,00	1,00
dantzig42	42	9	5	21,81	798,4	98560	25251	3600,34	798,4	27,76	1,00
eil51	51	11	5	74,52	489,6	126934	19953	3600,29	489,6	29,04	1,00
eil76	76	16	5	154,99	640,8	14054	1304	3600,17	640,8	34,58	1,00
fri26	26	6	5	9,04	956,6	132245	52766	3600,08	956,6	11,25	1,00
gr17	17	4	5	4,09	1997,2^{+}	40946	39517	218,52	1997,2*	0,00	1,00
gr21	21	5	5	5,47	3074	188649	85281	3600,05	3074	13,90	1,00
gr24	24	5	5	6,06	1469,6	232992	123440	3600,06	1469,6	6,21	1,00
gr48	48	10	5	53,47	5709,8	178206	27543	3600,29	5709,8	25,21	1,00
gr96	96	20	5	139,79	68446	13082	1089	3618,50	68446	40,46	1,00
hk48	48	10	5	44,46	13544,8	115071	22555	3600,41	13544,8	25,43	1,00
kroA100	100	20	5	297,29	28841,2	12933	1076	3600,49	28841,2	40,56	1,00
kroB100	100	20	5	272,25	28207,6	13254	1214	3600,50	28207,6	35,77	1,00
kroC100	100	20	5	418,10	27977,4	17925	1140	3627,84	27977,4	40,46	1,00
kroD100	100	20	5	284,65	28621	15845	1286	3600,35	28621	41,70	1,00
kroE100	100	20	5	210,63	29723,4	18965	1481	3600,09	29723,4	39,12	1,00
pr76	76	16	5	119,47	125243	18505	1995	3600,05	125243	37,66	1,00
rat99	99	20	5	141,74	1534,2	14203	1160	3605,13	1534,2	33,23	1,00
rd100	100	20	5	228,24	10523	15884	1318	3601,90	10523	37,26	1,00
st70	70	14	5	84,35	856,2	123620	12450	3600,30	856,2	27,52	1,00
swiss42	42	9	5	17,54	1476,2	136532	27996	3600,06	1476,2	25,46	1,00
ulysses16	16	4	4	4,59	7766,2^{+}	13605	11668	43,78	7766,2*	0,00	1,00
ulysses22	22	5	5	6,28	7165,6	233836	88613	3600,04	7165,6	16,46	1,00
Média	–	–	–	**104,47**	–	**87713,82**	**26366,43**	**3226,63**	–	**24,47**	**1,00**

Tabela A.10: Resultados para o cenário ST com $\alpha = 0{,}4$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	107,92	13226,2	138746	25400	3600,45	13226,2	25,68	1,00
bayg29	29	6	5	9,76	1916	181428	57623	3600,09	1916	11,83	1,00
bays29	29	6	5	9,18	2429,8	233591	82585	3600,05	2429,8	9,46	1,00
berlin52	52	11	5	63,97	9205,6	69606	14506	3600,45	9205,6	24,98	1,00
brazil58	58	12	5	144,47	28992	78723	13102	3600,18	28992	21,65	1,00
burma14	14	3	5	3,08	3891,4^{+}	8679	7390	26,15	3891,4*	0,00	1,00
dantzig42	42	9	5	18,17	868,6	121107	27322	3600,31	868,6	27,49	1,00
eil51	51	11	5	44,16	510,4	122748	19177	3600,32	510,4	26,49	1,00
eil76	76	16	5	70,98	676	13779	1447	3607,26	676	33,22	1,00
fri26	26	6	5	8,03	1038,2	111250	52655	3600,03	1038,2	9,59	1,00
gr17	17	4	5	4,59	2139,8^{+}	9880	9733	45,94	2139,8*	0,00	1,00
gr21	21	5	5	6,29	3314,4	127862	74488	3600,05	3314,4	11,95	1,00
gr24	24	5	5	7,17	1542,2	244475	128922	3600,08	1542,2	5,36	1,00
gr48	48	10	5	46,81	6038,4	163086	26638	3600,37	6038,4	24,77	1,00
gr96	96	20	5	194,16	71219,2	26884	2381	3600,46	71219,2	36,89	1,00
hk48	48	10	5	58,80	14358,4	115506	23441	3600,29	14358,4	24,92	1,00
kroA100	100	20	5	343,10	30771,4	13379	1402	3612,81	30771,4	37,52	1,00
kroB100	100	20	5	274,93	31092	15321	1293	3632,96	31092	38,28	1,00
kroC100	100	20	5	268,24	30911,4	20874	1357	3600,78	30911,4	42,05	1,00
kroD100	100	20	5	493,18	30517,6	15836	1272	3611,38	30517,6	39,61	1,00
kroE100	100	20	5	188,34	30975,6	13647	1062	3600,67	30975,6	38,44	1,00
pr76	76	16	5	105,17	136430	9707	1096	3600,97	136430	36,93	1,00
rat99	99	20	5	158,80	1646,6	14370	1222	3622,90	1646,6	34,42	1,00
rd100	100	20	5	291,69	11228,8	16692	1204	3607,24	11228,8	38,97	1,00
st70	70	14	5	71,50	889,2	102952	10794	3601,22	889,2	28,74	1,00
swiss42	42	9	5	40,65	1570,4	154541	30644	3600,20	1570,4	23,97	1,00
ulysses16	16	4	4	5,00	8073,4^{+}	13693	11915	37,11	8073,4*	0,00	1,00
ulysses22	22	5	5	6,64	7544,2	156102	81071	3600,11	7544,2	8,03	1,00
Média	–	–	–	**108,74**	–	**82659,43**	**25397,93**	**3221,82**	–	**23,62**	**1,00**

Tabela A.11: Resultados para o cenário ST com $\alpha = 0{,}6$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	120,54	14185,2	159583	27728	3600,44	14185,2	25,56	1,00
bayg29	29	6	5	9,71	2014	193196	63617	3601,32	2014	10,22	1,00
bays29	29	6	5	9,15	2545,2	196541	71869	3600,45	2545,2	9,86	1,00
berlin52	52	11	5	114,92	9802,2	83441	17000	3600,36	9802,2	24,25	1,00
brazil58	58	12	5	303,85	31213,8	87610	14069	3600,44	31213,8	21,39	1,00
burma14	14	3	5	3,06	$4102{,}6^{+}$	12275	10368	26,70	4102,6*	0,00	1,00
dantzig42	42	9	5	37,42	938,8	123332	29766	3600,35	938,8	27,52	1,00
eil51	51	11	5	46,45	546,6	128051	19958	3600,19	546,6	27,95	1,00
eil76	76	16	5	78,03	709,6	21439	2043	3600,08	709,6	31,93	1,00
fri26	26	6	5	11,62	1119,8	115752	60468	3600,07	1119,8	10,06	1,00
gr17	17	4	5	4,50	$2270{,}2^{+}$	6424	6399	28,02	2270,2*	0,00	1,00
gr21	21	5	5	6,24	3518,6	136565	96369	3600,06	3518,6	8,06	1,00
gr24	24	5	5	6,90	1614,8	225353	104358	3600,02	1614,8	6,56	1,00
gr48	48	10	5	58,50	6394,8	154471	27099	3600,06	6394,8	24,96	1,00
gr96	96	20	5	566,62	76272	23388	2088	3616,16	76272	36,14	1,00
hk48	48	10	5	57,13	15225,2	109934	25641	3600,29	15225,2	24,96	1,00
kroA100	100	20	5	249,15	32796,2	16025	1177	3610,92	32796,2	40,90	1,00
kroB100	100	20	5	245,93	33511,6	10411	1147	3635,71	33511,6	38,83	1,00
kroC100	100	20	5	588,54	32114,8	15700	1215	3613,99	32114,8	39,37	1,00
kroD100	100	20	5	287,52	32048	13955	1218	3609,38	32048	37,98	1,00
kroE100	100	20	5	268,93	33559,8	16603	1327	3606,19	33559,8	40,10	1,00
pr76	76	16	5	112,49	142362	18849	2098	3600,60	142362	34,07	1,00
rat99	99	20	5	159,60	1761,6	15406	1240	3632,48	1761,6	36,29	1,00
rd100	100	20	5	350,12	11766,4	16781	1234	3600,72	11766,4	38,19	1,00
st70	70	14	5	110,66	951	109173	11435	3601,15	951	32,20	1,00
swiss42	42	9	5	25,07	1661,2	132016	29183	3600,25	1661,2	23,51	1,00
ulysses16	16	4	4	4,86	$8344{,}6^{+}$	18848	13867	47,91	8344,6*	0,00	1,00
ulysses22	22	5	5	6,66	7849,8	157410	112306	3600,05	7849,8	3,47	1,00
Média	–	–	–	**137,29**	–	**82804,71**	**27010,25**	**3222,66**	–	**23,37**	**1,00**

Tabela A.12: Resultados para o cenário ST com $\alpha = 0{,}8$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	5	57,34	15175,6	129876	30593	3600,46	15175,6	27,02	1,00
bayg29	29	6	5	9,54	2112	136379	51507	3600,07	2112	14,62	1,00
bays29	29	6	5	10,07	2654,4	192390	62166	3600,03	2654,4	10,31	1,00
berlin52	52	11	5	107,51	10340,8	82240	18713	3600,06	10340,8	25,77	1,00
brazil58	58	12	5	129,81	33213,2	88277	15641	3600,25	33213,2	22,96	1,00
burma14	14	3	5	3,04	4270,8^{+}	6086	5702	35,68	4270,8*	0,00	1,00
dantzig42	42	9	5	36,13	1005,4	145510	31620	3600,21	1005,4	27,23	1,00
eil51	51	11	5	57,60	565,6	99523	23864	3600,47	565,6	27,12	1,00
eil76	76	16	5	94,91	746,2	16562	1711	3600,81	746,2	30,96	1,00
fri26	26	6	5	8,06	1201,4	112021	56355	3600,09	1201,4	11,05	1,00
gr17	17	4	5	4,54	2392,6^{+}	8723	7925	32,23	2392,6*	0,00	1,00
gr21	21	5	5	6,16	3715,8	158784	120707	3600,02	3715,8	6,35	1,00
gr24	24	5	5	7,04	1676,6	280066	116661	3600,13	1676,6	5,93	1,00
gr48	48	10	5	64,91	6747,8	153283	25871	3600,22	6747,8	25,50	1,00
gr96	96	20	5	353,49	80701	18321	1765	3617,83	80701	35,91	1,00
hk48	48	10	5	63,42	15996,4	118391	22726	3600,36	15996,4	25,00	1,00
kroA100	100	20	5	337,85	34393,4	15688	1282	3626,01	34393,4	40,58	1,00
kroB100	100	20	5	376,33	35024,6	14392	1189	3605,10	35024,6	39,13	1,00
kroC100	100	20	5	206,77	33982,6	16481	1196	3601,40	33982,6	41,12	1,00
kroD100	100	20	5	354,27	34088,6	14518	1111	3631,67	34088,6	39,69	1,00
kroE100	100	20	5	268,50	35271,2	17635	1325	3630,23	35271,2	40,09	1,00
pr76	76	16	5	78,05	152492	11350	1245	3600,04	152492	34,12	1,00
rat99	99	20	5	261,43	1847,6	10169	1157	3607,38	1847,6	36,72	1,00
rd100	100	20	5	191,04	12193,4	14190	1128	3600,20	12193,4	37,41	1,00
st70	70	14	5	89,73	998,8	11242	1402	3600,31	998,8	31,20	1,00
swiss42	42	9	5	29,55	1754,2	143377	29435	3600,09	1754,2	24,64	1,00
ulysses16	16	4	4	4,97	8615,8^{+}	9859	7250	28,42	8615,8*	0,00	1,00
ulysses22	22	5	5	6,71	8155,4	214811	124533	3600,02	8155,4	2,45	1,00
Média	–	–	–	**114,96**	–	**80005,14**	**27349,29**	**3222,14**	–	**23,67**	**1,00**

Tabela A.13: Resultados para o cenário SL com $\alpha = 0{,}2$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	103,68	10346,2	106623	28965	3600,46	10346,2	21,56	1,00
bayg29	29	6	9	17,37	1539,4	100255	48776	3600,02	1539,4	12,59	1,00
bays29	29	6	9	19,85	1891,4	103195	46784	3600,12	1891,4	11,93	1,00
berlin52	52	11	9	141,30	6985,6	81146	18627	3600,75	6985,6	22,64	1,00
brazil58	58	12	9	131,55	22602,2	18143	3767	3600,09	22602,2	33,28	1,00
burma14	14	3	9	2,32	2832^{+}	1889	2034	1,47	2832*	0,00	1,00
dantzig42	42	9	9	27,18	677	73595	26827	3600,46	677	20,59	1,00
eil51	51	11	9	71,95	427	67162	17025	3600,56	427	20,61	1,00
eil76	76	16	9	122,02	550,4	17433	2420	3600,04	550,4	26,56	1,00
fri26	26	6	9	16,52	829,4	123876	62859	3600,08	808,8	11,42	1,03
gr17	17	4	9	4,18	$1756{,}4^{+}$	129982	135166	3229,90	1756,4*	0,00	1,00
gr21	21	5	9	6,73	2460	132115	92370	3600,06	2460	7,95	1,00
gr24	24	5	9	7,87	1192	73450	66975	3600,07	1192	5,50	1,00
gr48	48	10	9	44,23	4885,4	87951	27662	3600,56	4885,4	21,52	1,00
gr96	96	20	9	350,77	57962,2	47415	5497	3600,17	57962,2	36,34	1,00
hk48	48	10	9	76,38	11116,2	69455	21475	3600,50	11015,6	16,73	1,01
kroA100	100	20	9	308,39	23414	27216	3665	3625,26	23414	31,92	1,00
kroB100	100	20	9	438,01	24124,4	26170	3307	3637,95	24124,4	31,23	1,00
kroC100	100	20	9	397,66	23424,4	29319	3586	3600,14	23424,4	33,21	1,00
kroD100	100	20	9	525,28	23178,6	22578	3254	3600,76	23178,6	31,27	1,00
kroE100	100	20	9	471,44	23887	19748	2678	3600,06	23887	30,84	1,00
pr76	76	16	9	161,29	109929	28998	4627	3600,29	109929	32,16	1,00
rat99	99	20	9	764,12	1311,6	38129	3990	3600,16	1311,6	26,03	1,00
rd100	100	20	9	483,85	8660,6	27968	3237	3625,97	8660,6	31,54	1,00
st70	70	14	9	108,98	687,4	13523	2246	3604,95	687,4	26,91	1,00
swiss42	42	9	9	31,91	1314,2	98939	32152	3600,27	1282	21,81	1,03
ulysses16	16	4	7	3,72	4676^{+}	6617	8106	23,92	4676*	0,00	1,00
ulysses22	22	5	9	7,52	$4433{,}6^{+}$	47577	47828	332,76	4433,6*	0,00	1,00
Média	–	–	–	**173,07**	–	**57873,82**	**25925,18**	**3217,42**	–	**20,22**	**1,00**

Tabela A.14: Resultados para o cenário SL com $\alpha = 0{,}4$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	167,32	11259,6	68419	24408	3600,09	11045,8	18,98	1,02
bayg29	29	6	9	15,69	1640,6	66397	46841	3600,09	1640,6	10,30	1,00
bays29	29	6	9	16,47	2041,8	79384	45317	3600,05	2041,8	10,24	1,00
berlin52	52	11	9	82,36	7987,2	77515	17482	3600,35	7987,2	23,60	1,00
brazil58	58	12	9	206,41	25105	53697	12423	3600,28	25105	25,33	1,00
burma14	14	3	9	2,58	3136,4^{+}	2870	3500	12,83	3136,4*	0,00	1,00
dantzig42	42	9	9	43,06	738,2	64278	26111	3600,36	738,2	18,94	1,00
eil51	51	11	9	69,09	460	79105	21401	3600,99	455	19,96	1,01
eil76	76	16	9	138,16	596,6	20167	2794	3600,38	596,6	25,95	1,00
fri26	26	6	9	9,73	906,8	87592	60055	3600,08	906,8	8,82	1,00
gr17	17	4	9	4,44	1977,8^{+}	13247	14604	62,54	1977,8*	0,00	1,00
gr21	21	5	9	6,12	2754	113291	95619	3600,05	2754	4,67	1,00
gr24	24	5	9	9,65	1287	63012	70374	3600,11	1287	4,21	1,00
gr48	48	10	9	71,41	5324,2	82638	28121	3600,83	5324,2	20,27	1,00
gr96	96	20	9	363,17	64178,2	36005	4558	3602,76	64178,2	34,34	1,00
hk48	48	10	9	57,72	12261,8	76864	23732	3600,58	12261,8	18,71	1,00
kroA100	100	20	9	493,93	24694,8	18467	2977	3600,48	24694,8	29,58	1,00
kroB100	100	20	9	493,86	25042	22692	3209	3638,83	25042	27,54	1,00
kroC100	100	20	9	492,32	25456,6	23379	3069	3600,17	25456,6	33,36	1,00
kroD100	100	20	9	425,03	24846	26871	3721	3608,56	24846	29,62	1,00
kroE100	100	20	9	900,60	25416	25083	3049	3606,94	25416	29,13	1,00
pr76	76	16	9	288,81	122541	21367	3085	3600,05	122541	31,63	1,00
rat99	99	20	9	379,06	1398,8	21041	2628	3601,35	1398,8	25,86	1,00
rd100	100	20	9	767,68	9259	52642	6469	3600,17	9259	30,15	1,00
st70	70	14	9	208,31	752,2	14946	2467	3604,79	750,2	26,13	1,00
swiss42	42	9	9	41,26	1375,6	87200	28247	3600,15	1375,6	20,62	1,00
ulysses16	16	4	7	3,99	5588^{+}	2761	3402	13,26	5588*	0,00	1,00
ulysses22	22	5	9	8,19	5348,2^{+}	10810	10709	95,80	5348,2*	0,00	1,00
Média	–	–	–	**205,94**	–	**46847,86**	**20370,43**	**3094,75**	–	**18,86**	**1,00**

Tabela A.15: Resultados para o cenário SL com $\alpha = 0{,}6$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	112,04	12052	57169	25655	3600,40	12052	19,84	1,00
bayg29	29	6	9	12,90	1712,4	64063	55043	3601,19	1712,4	7,40	1,00
bays29	29	6	9	23,09	2149	64277	55338	3600,10	2149	7,58	1,00
berlin52	52	11	9	120,72	8394,6	55563	16957	3600,46	8394,6	18,43	1,00
brazil58	58	12	9	132,60	27948,4	69157	19696	3600,24	27948,4	23,30	1,00
burma14	14	3	9	2,62	3370,2^{+}	2171	3459	11,36	3370,2*	0,00	1,00
dantzig42	42	9	9	31,25	795,8	64199	34132	3600,41	795,8	18,87	1,00
eil51	51	11	9	59,78	480,4	61745	22899	3600,87	480,4	19,53	1,00
eil76	76	16	9	153,82	630,8	15972	2554	3607,83	630,8	24,45	1,00
fri26	26	6	9	16,16	1003,6	75351	54704	3600,19	1003,6	8,69	1,00
gr17	17	4	9	4,51	2066,6^{+}	2604	3695	20,47	2066,6*	0,00	1,00
gr21	21	5	9	6,43	3048	114379	102993	3600,06	3023	5,80	1,01
gr24	24	5	9	7,69	1375	72841	75026	3600,10	1375	5,43	1,00
gr48	48	10	9	59,46	5654,8	77243	24895	3600,51	5630,4	18,79	1,00
gr96	96	20	9	489,54	68027,2	45450	5942	3615,51	68027,2	31,25	1,00
hk48	48	10	9	94,77	12996	78405	27447	3600,33	12996	18,19	1,00
kroA100	100	20	9	1162,97	27076,6	39809	5619	3612,25	26805,6	29,92	1,01
kroB100	100	20	9	578,49	27416,8	30594	3544	3600,16	27416,8	28,73	1,00
kroC100	100	20	9	820,54	26970,8	46398	5737	3650,68	26970,8	32,25	1,00
kroD100	100	20	9	442,89	27257,8	40790	5330	3643,85	27257,8	30,76	1,00
kroE100	100	20	9	473,29	27826,6	25042	3738	3615,98	27826,6	30,17	1,00
pr76	76	16	9	433,16	129503	15925	2568	3603,11	129503	28,93	1,00
rat99	99	20	9	556,12	1524,4	34217	3987	3600,25	1524,4	28,06	1,00
rd100	100	20	9	493,00	9916	30941	4319	3608,10	9916	29,42	1,00
st70	70	14	9	159,18	801,4	9124	2099	3600,64	801,4	25,01	1,00
swiss42	42	9	9	28,82	1507	71996	35191	3600,28	1507	21,55	1,00
ulysses16	16	4	7	3,95	6462,4^{+}	3360	4052	26,81	6462,4*	0,00	1,00
ulysses22	22	5	9	7,68	6262,8^{+}	3577	4046	62,73	6262,8*	0,00	1,00
Média	–	–	–	**231,70**	–	**45441,50**	**21809,46**	**3095,89**	–	**18,30**	**1,00**

Tabela A.16: Resultados para o cenário SL com $\alpha = 0{,}8$, com ciclos degenerados.

Instância	n	K	C	Heurística		Branch-and-cut					Razão
				tempo	valor	cortes	nós	tempo	valor	Gap (%)	
att48	48	10	9	158,78	12762,2	45546	29060	3600,90	12570,2	17,89	1,02
bayg29	29	6	9	13,66	1784,2	53918	56230	3600,17	1784,2	7,02	1,00
bays29	29	6	9	17,00	2239,2	52031	56188	3600,18	2212,2	6,23	1,01
berlin52	52	11	9	138,97	8963,2	47228	21554	3600,93	8963,2	17,58	1,00
brazil58	58	12	9	163,81	30182	48173	15162	3600,58	30182	21,42	1,00
burma14	14	3	9	2,59	3509^{+}	1800	1746	2,17	3509*	0,00	1,00
dantzig42	42	9	9	29,74	850,4	55070	33264	3600,40	850,4	19,76	1,00
eil51	51	11	9	106,29	506,2	53493	25290	3600,25	506,2	19,60	1,00
eil76	76	16	9	196,86	661,6	16493	2553	3600,09	661,6	23,10	1,00
fri26	26	6	9	13,38	1071	72142	61674	3600,06	1071	7,54	1,00
gr17	17	4	9	4,52	$2128{,}8^{+}$	506	517	1,35	2128,8*	0,00	1,00
gr21	21	5	9	6,25	3185,2	145769	134384	3600,06	3185,2	4,51	1,00
gr24	24	5	9	12,30	1452,4	84856	69897	3600,06	1452,4	6,55	1,00
gr48	48	10	9	80,00	5905	62600	27960	3600,65	5905	17,90	1,00
gr96	96	20	9	553,67	72693	43190	6215	3624,88	72693	30,04	1,00
hk48	48	10	9	125,52	13583,4	51274	23562	3600,29	13583,4	17,30	1,00
kroA100	100	20	9	662,54	28356,2	22176	3525	3600,51	28356,2	29,22	1,00
kroB100	100	20	9	531,50	29788	28405	4294	3600,70	29788	30,20	1,00
kroC100	100	20	9	769,84	28418,6	19151	4263	3627,23	28418,6	31,28	1,00
kroD100	100	20	9	1011,75	28720,6	24062	5529	3603,16	28720,6	30,03	1,00
kroE100	100	20	9	573,74	29645	25933	3674	3626,40	29645	30,18	1,00
pr76	76	16	9	344,41	137126	12793	2288	3601,61	137126	27,33	1,00
rat99	99	20	9	837,21	1598,8	26660	4167	3613,21	1598,8	27,68	1,00
rd100	100	20	9	391,94	10658	19741	3179	3659,55	10427	28,30	1,02
st70	70	14	9	123,81	854,8	8162	1758	3600,90	854,8	25,22	1,00
swiss42	42	9	9	39,80	1559	63132	35936	3600,66	1559	20,13	1,00
ulysses16	16	4	7	4,02	$6987{,}8^{+}$	822	928	1,44	6987,8*	0,00	1,00
ulysses22	22	5	9	7,51	$7167{,}4^{+}$	8676	9420	78,94	7167,4*	0,00	1,00
Média	–	–	–	**247,19**	–	**39064,36**	**23007,75**	**3094,55**	–	**17,71**	**1,00**