UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

ANDERSSON ALVES DA SILVA

UMA ABORDAGEM HEURÍSTICA PARA O PROBLEMA DO CARTEIRO CHINÊS CAPACITADO NA COLETA DE LIXO URBANO

Recife
2020

ANDERSSON ALVES DA SILVA

UMA ABORDAGEM HEURÍSTICA PARA O PROBLEMA DO CARTEIRO CHINÊS CAPACITADO NA COLETA DE LIXO URBANO

Dissertação de Mestrado apresentada à UFPE para a obtenção de grau de Mestre como parte das exigências do Programa de Pós-Graduação em Engenharia de Produção.

Área de Concentração: Pesquisa Operacional

Orientador: Prof. PhD Sóstenes Luiz Soares Lins.

Recife

2020

Catalogação na fonte
Bibliotecário Gabriel Luz, CRB-4 / 2222

S586a Silva, Andersson Alves da.
Uma abordagem heurística para o problema do carteiro chinês capacitado na coleta de lixo urbano / Andersson Alves da Silva – Recife, 2020.
101 f.: figs., quads., tabs., abrev. e siglas.

Orientador: Prof. Dr. Sóstenes Luiz Soares Lins.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia de Produção, 2020.
Inclui referências.

1. Engenharia de Produção. 2. Problema do carteiro chinês capacitado. 3. Coleta de lixo urbano. 4. Heurística. Roteamento de arcos capacitados. I. Lins, Sóstenes Luiz Soares (Orientador). II. Título.

UFPE

658.5 CDD (22. ed.) BCTG / 2021-10

ANDERSSON ALVES DA SILVA

UMA ABORDAGEM HEURÍSTICA PARA O PROBLEMA DO CARTEIRO CHINÊS CAPACITADO NA COLETA DE LIXO URBANO

Dissertação de Mestrado apresentada ao Departamento de Engenharia de Produção da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Engenharia de Produção.

Aprovada em: 02/03/2020.

BANCA EXAMINADORA

_____

Prof. PhD Sóstenes Luiz Soares Lins (Orientador)
Universidade Federal de Pernambuco

_____

Prof. Dr. Raphael Harry Frederico Ribeiro Kramer (Examinador Interno)
Universidade Federal de Pernambuco

_____

Prof. PhD Nivan Roberto Pereira Júnior (Examinador Externo)
Universidade Federal de Pernambuco

AGRADECIMENTOS

Diante de dois anos de muitos desafios e trabalho, mas também muito conhecimento adquirido e grandes aprendizados, tenho tanto a agradecer...

Primeiramente, ao Pai Celestial, Deus, que ouviu minhas orações quando sempre precisei. Por me fazer entender que diante do Seu silêncio, eu deveria ter paciência, que tudo se resolveria, e que quando eu não aguentava mais, soprou nos meus ouvidos que estava tudo bem, porque o que realmente importa é eu ter dado o meu melhor dia após dia.

Agradecer imensamente a minha família maravilhosa, aqueles que aceitaram esta minha jornada fora de casa, longe de seus olhos, de seus abraços e de seus conselhos e que me deram o maior apoio do mundo: a confiança de que estou trilhando o caminho certo e entendem que o meu afastamento foi importante para o objetivo maior. Minha mãe Maria Aparecida da Silva, meu pai Francisco Hélio Alves da Silva, meus irmãos Ailton Cezar Alves da Silva e Ronniel Alves da Silva, meus sobrinhos Andreza e Heitor, minha cunhada Jaiane Silva e meus queridos e amados parentes que tanto se orgulham do caminho que venho trilhando. Amo vocês.

À minha querida e amada Amanda Silva, que fez desta experiência única, uma estrada mais fácil de andar e desfrutar deste novo ambiente chamado “pesquisa”. Grato por todos os seus conselhos e broncas para que eu pudesse melhorar pessoalmente e profissionalmente nesses últimos dois anos. Certamente, nós estamos no caminho certo. Obrigado por existir.

Ao incrível orientador que tive, professor Sóstenes Luiz Soares Lins, que foi um segundo pai para mim, enquanto eu estive em Recife. Sempre atencioso e com um bom humor, que faze toda diferença na hora de se trabalhar junto. Tenho muito orgulho em chamá-lo de professor, orientador, amigo, inspirador. E mais orgulho ainda de conhecer um dos maiores matemáticos brasileiros e compartilhar com ele um pouco do seu imenso conhecimento durante esses dois anos de mestrado.

Agradecer aos meus, e minhas, colegas do Programa de Pós-Graduação em Engenharia de Produção (PPGEP). Estudar o conteúdo das disciplinas com vocês facilitou muito o aprendizado, além de tantos momentos de descontração que fazem valer a pena todo o esforço gasto. Fico grato em compartilhar um pouco do que aprendi com pessoas tão incríveis, tão geniais e ao mesmo tempo tão humildes. Com certeza toda a turma do PPGEP 2018 estará marcada para sempre na minha memória.

Aos professores e professoras do PPGEP por toda a atenção aos alunos do programa, por todo conhecimento transmitido e terem essa didática incrível, que facilita e muito o

aprendizado. Em especial, agradeço aos professores: Isis Didier Lins, Márcio José das Chagas Moura, Adiel Teixeira de Almeida Filho e Danielle Costa Morais.

Um agradecimento especial à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) por todo o apoio e financiamento durante todo o mestrado, sem este fomento seria impossível a realização de todos os trabalhos desenvolvidos.

Aos professores Raphael Harry Frederico Ribeiro Kramer e Nivan Roberto Ferreira Júnior, que participaram da minha banca de defesa do Mestrado Acadêmico. Eu agradeço por todas as contribuições e sugestões dadas para a melhoria contínua do trabalho.

Ao Centro de Informática (CIn) da UFPE por me receber bem em seus laboratórios e instalações pelos dois anos de mestrado e por fazer o meu interesse pela área da computação só aumentar com o tempo. Certamente o CIn-UFPE será a minha melhor casa nos próximos 4 anos de Doutorado.

À empresa Autarquia de Manutenção e Limpeza Urbana do Recife (EMLURB) por ter cedido os dados para aplicação no programa desenvolvido e ao Sr. Antônio Avelino, responsável pela administração de toda a parte de saneamento básico na cidade de Recife-PE, pela entrevista pessoal em seu escritório para tirar diversas dúvidas sobre o sistema de coleta de lixo da cidade.

Ao mais, a todos que não foram citados e contribuíram de forma direta ou indireta para a conclusão deste trabalho incrível.

“Quando a vida nos deixa cegos. O amor nos mantém gentis”.

(The Messenger, Linkin Park).

RESUMO

A problemática da quantidade de Resíduos Sólidos Urbanos (RSU) gerados representa um dos maiores desafios para os gestores públicos nas grandes cidades. Para tanto, a atividade da coleta dos RSU possui diversas decisões importantes, que deverão ser tomadas para gerar rotas eficientes e por este motivo, a coleta é a operação mais importante dentro da cadeia produtiva do lixo. Para esta resolução, utilizou-se a otimização de rotas pelo conhecido Problema do Carteiro Chinês Capacitado (PCCC), que gera rotas viáveis através da cobertura de todos os segmentos de ruas respeitando a capacidade dos veículos coletores. É um problema cuja resolução requer muito tempo computacional. Uma abordagem heurística é capaz de gerar soluções melhores do que as empíricas, que podem ser analisadas em um curto intervalo de tempo para a tomada de decisão. O trabalho objetiva propor um novo procedimento heurístico para resolução do PCCC em 4 passos, onde inicialmente se resolve o problema por uma abordagem exata desconsiderando as demandas dos arcos, depois é traçado uma única rota percorrendo todos os arcos do grafo, e posteriormente, quebra-se a rota em várias respeitando a capacidade máxima por veículo. A abordagem foi implementada e aplicada para dois bairros na cidade de Recife-PE e para três instâncias onde se comparou a solução heurística com uma solução exata descrito por um modelo matemático de Golden e Wong (1981) para analisar o valor da função objetivo e seu tempo de resolução. Para a aplicação nos dois bairros (Engenho do Meio e Cordeiro) 10 resultados foram obtidos para cada e verificado qual apresentou melhor solução. Quanto as instâncias, a capacidade máxima do veículo foi variada e a resolução da heurística proposta foi comparada com o modelo exato utilizado. Bons resultados foram alcançados quando a capacidade máxima do veículo aumentava.

Palavras-chave: Problema do carteiro chinês capacitado. Coleta de lixo urbano. Heurística. Roteamento de arcos capacitados.

ABSTRACT

The problem of the amount of Solid Urban Waste (SUW) generated represents one of the biggest challenges for public managers in large cities. Therefore, the SUW collection activity has several important decisions that must be taken to generate efficient routes and for this reason, the collection is the most critical operation within the waste production chain. For this resolution, the optimization of routes was used by the well-known Capacitated Chinese Postman Problem (CCPP), which generates viable routes through the coverage of all street segments while respecting the capacity of the collecting vehicles. It is a problem whose resolution requires a lot of computational time. A heuristical approach is able to generate better solutions than empirical ones. They can be analyzed in a short time for decision making. The study aims to describe a heuristic procedure for solving the CCPP in 4 steps, where the problem is initially solved by an exact approach disregarding the demands of the arcs, then a single route is traced through all the arcs of the graph, and later, the route is broken in several respecting the maximum capacity per vehicle. The approach was implemented and applied to two neighborhoods in the city of Recife-PE and to three instances where the heuristic solution was compared with an exact solution described by a mathematical model by Golden and Wong (1981) to analyze the value of the objective function and its resolution time. For the application in the two neighborhoods (Engenho do Meio and Cordeiro) 10 results were obtained for each one and it was verified which one presented the best solution. As for the instances, the maximum capacity of the vehicle was varied and the resolution of the proposed heuristic was compared with the exact model used. Good results were achieved when the maximum capacity of the vehicle was increased.

Keywords: Capacitated Chinese postman problem. Urban garbage collection. Capacitated arcs routing.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

A	Conjunto de Arcos de um grafo
E	Conjunto de Arestas de um grafo
G	Grafo
MPL	Modelo de Programação Linear
N	Conjunto de nós/vértices de um grafo
NP	*Nondeterministic polynomial time*
NP-completo	*Nondeterministic polynomial time complete*
P	*Polynomial*
PCC	Problema do Carteiro Chinês
PCCC	Problema do Carteiro Chinês Capacitado
PCCD	Problema do Carteiro Chinês Direcionado
PCCM	Problema do Carteiro Chinês Misto
PCCND	Problema do Carteiro Chinês Não Direcionado
PCCV	Problema do Carteiro Chinês Com Vento
PCR	Problema do Carteiro Rural
PCV	Problema do Caixeiro Viajante
PLIM	Programação Linear Inteira Mista
PRA	Problema de Roteamento em Arcos
PRAC	Problema de Roteamento de Arcos Capacitados
PRACM	Problema de Roteamento de Arcos Capacitados Mistos
PRL	Problema de Roteamento de Leituristas
PRV	Problema de Roteamento de Veículos
PRVC	Problema de Roteamento de Veículos Capacitados
PRVCR	Problema de Roteamento de Veículos para Coleta de Resíduos
RSU	Resíduos Sólidos Urbanos

SUMÁRIO

1 INTRODUÇÃO

O presente capítulo introduz a problemática acerca dos resíduos sólidos e sua abordagem para resolução por meio de Problemas de Roteamento em Arcos e o Problema do Carteiro Chinês Capacitado. Em seguida, são definidos os objetivos geral e específicos do trabalho e a justificativa para desenvolvimento deste estudo.

1.1 DESCRIÇÃO DO PROBLEMA

Segundo Jacinto, Rosa e Banos (2014) um dos maiores problemas enfrentados nas grandes cidades está ligado ao aumento na formação de Resíduos Sólidos Urbanos (RSU), sendo esta uma consequência direta do crescimento populacional. Os autores ainda sugerem que há a necessidade da existência de políticas públicas eficientes para a realização da coleta, o transporte e a destinação destes resíduos. Nesta perspectiva, a problemática da quantidade de RSU gerados representa um dos grandes desafios para os gestores públicos (SOUZA; GONÇALVES; CURI, 2018).

A realização da coleta dos RSU nas cidades é de responsabilidade dos órgãos públicos municipais (MORO et al., 2018a) que, muitas vezes, terceirizam esta atividade para empresas especializadas, que ainda partem do conhecimento profissional de seus funcionários para realizar a melhor rota (ou percurso) para a coleta destes resíduos. Para tanto, a realização desta atividade deve ser a mais bem planejada, por representar em torno de 50% até 80% do custo de operação de limpeza pública (VECCHI et al., 2015).

Segundo pesquisa da ABRELPE (2019), no ano de 2018 o Brasil gerou um total de 79 milhões de toneladas de RSU e foi coletado cerca de 72,7 milhões de toneladas, sendo que 6,3 milhões de toneladas dos resíduos tiveram possivelmente destino impróprio. A pesquisa afirma que a geração de RSU atingiu um total de 216.629 toneladas diárias, representando um aumento de quase 1% em referência ao ano anterior. Este aumento refletiu em um custo de R$ 10,15 por habitante por mês, para a realização de todos os serviços de limpeza urbana no país.

A atividade da coleta de RSU possui diversas decisões importantes que deverão ser tomadas, como por exemplo, a abertura de uma nova instalação para atender a demanda de coletas, a alocação dos veículos e, obviamente, rotas eficientes (BELIËN; DE BOECK; VAN ACKERE, 2012). Os autores ainda ressaltam que a coleta é a operação mais importante e cara do ciclo do descarte dos RSU por causa da intensidade do trabalho e do uso dos caminhões no processo da coleta.

Por se tratar de uma atividade de grande importância na cadeia produtiva, a coleta de resíduos pode ser considerada uma atividade logística bastante relevante para qualquer cidade, cuja coleta residencial deve proceder da forma mais econômica (GRAKOVA et al., 2018; MASRAN; RAMLI, 2019). Por conseguinte, a coleta dos RSU depende muito dos problemas de otimização de rotas, por envolver uma quantidade alta de gastos em termos de recursos financeiros, mão-de-obra e custos operacionais variáveis que precisam ser minimizados. Portanto, quanto melhor for a rota da coleta dos resíduos, maior será a redução nos custos, como também, dos impactos ambientais gerados (HANNAN, 2018; MASRAN; RAMLI, 2019). Os efeitos ambientais surgem, neste contexto, com o alto consumo de combustível, emissão de gás carbônico do veículo coletor, além de gerar o acúmulo de lixo em ruas que não foram atendidas.

A otimização das rotas é estudada pelos Problemas de Roteamento, que são separados em problemas em arcos e problemas em nós (WØHLK; LAPORTE, 2018). Os Problemas de Roteamento em Arcos, em especial o Problema do Carteiro Chinês (PCC), otimizam rotas a partir da cobertura de todos os arcos (ruas), tornando a coleta de RSU possível e reduzindo os gastos com custos operacionais.

O PCC, entre várias aplicações, é tratado na literatura para resolver diversos problemas de roteamento de arcos, como: rota de carteiros (IRNICH, 2008); rotas para leituristas de ruas (USBERTI et al., 2008; SILVA; LINS; XAVIER, 2019a); leituristas de barragens (DOS SANTOS, 2016); patrulha de segurança nas ruas (SHAFAHI; HAGHANI, 2015); leitura de medidores de energia elétrica (STERN; DROR, 1979); inspeção e manutenção de linhas ferroviárias (YILMAZ; ÇODUR; YILMAZ, 2017); patrulhamento estratégico de seguranças (HOCHBAUM; LYU; ORDÓÑEZ, 2014; SHAFAHI; HAGHANI, 2015); serviços de inverno (remoção de neve, derramamento de sal em estradas, etc.) (LUKMAN et al., 2018); fiscalização de linhas de ônibus; serviços de transporte escolar, etc.

Uma variação do PCC encontrada na literatura inclui restrições da quantidade de veículos e a capacidade de cada um, de modo que coletem os resíduos nos diversos bairros de uma cidade percorrendo a menor distância possível. Este problema é chamado de Problema do Carteiro Chinês Capacitado (PCCC) e devido sua complexidade ser *NP-difícil* (TING; TSAI, 2018), sua resolução requer muito tempo computacional para aplicações reais.

Em uma abordagem por meio de heurísticas, embora não garanta a otimalidade, mas poderá encontrar soluções melhores do que as empíricas. Além disso, podem ser avaliadas em um curto intervalo de tempo, facilitando aos tomadores de decisão a possiblidade de compararem as alternativas geradas e garantir às empresas de logística a obtenção de rotas eficazes, com um menor custo na entrega (MORO et al., 2015; MORO et al., 2018b).

Entre as heurísticas para resolução dos Problemas de Roteamento em Arcos, as mais simples são os chamados métodos construtivos (LENIS; RIVERA, 2018), que podem ser divididos, segundo Prins, Labadi e Reghioui (2009) em quatro tipos: (1) Inserção, criam rotas uma a uma por inserções sucessivas de tarefas; (2) Mesclagem, criam uma rota inicial para cada tarefa e depois mesclam rotas em pares; (3) *Cluster-First, Route-Second*, definem um *cluster* de tarefas para cada veículo e, em seguida, criam uma rota em cada *cluster*; e (4) *Route-First Cluster-Second* ou *Tour Splitting*, calculam um *tour* gigante (*giant tour*) relaxando a restrição de capacidade e, em seguida, dividem esse *tour* em rotas com capacidade viável.

Este último tipo (*Route-First Cluster-Second*), de acordo com Van Bevern, Komusiewicz e Sorge (2017) é a única abordagem conhecida para resolução de Problemas de Roteamento em Arcos a fornecer soluções de qualidade em tempo polinomial.

Diante do contexto, o presente trabalho apresenta um estudo aprofundado sobre o PCC, com ênfase no PCCC e descreve um procedimento do tipo *Route-First Cluster-Second* para resolução do PCCC com foco na coleta de lixo em 4 passos: escrever um arquivo com um modelo exato para o PCC, resolvê-lo por um *solver* eficiente, sequenciar o percurso em uma única rota (*tour*) e dividi-lo em *m* rotas por uma nova abordagem heurística, de modo que as rotas dos veículos sejam designadas para melhorar a coleta do lixo urbano.

O estudo foi implementado em linguagem de programação *Python*, aplicado em um caso real na cidade de Recife-PE e para algumas instâncias e, por fim, comparado a um modelo de Programação Linear Inteira Mista conhecido na literatura.

1.2 JUSTIFICATIVA

O presente trabalho se justifica por diversos fatores, sendo aqui explorados três mais importantes para as áreas socioambiental, econômico e científico.

As diversas aplicações de roteamento de arcos encontrados na literatura abordam a questão econômica. Tendo conhecimento que a utilização dos recursos públicos para a operação de limpeza urbana é considerada crítica no Brasil ABRELPE (2019), então toda forma de otimização eficiente para reduzir custos operacionais é de grande interesse para os gestores públicos. A verba que não é gasta nesses serviços, é um recurso que poderá ser usado em outras áreas consideradas de prioridade.

O desperdício de resíduos é uma questão que tem causado grande preocupação pública nas sociedades modernas, não apenas pelo aumento da quantidade de lixo gerado, mas também pela crescente complexidade de alguns produtos e componentes (HAN; PONCE CUETO,

2015). Sendo assim, uma coleta de lixo efetiva contribuirá para o controle desse desperdício em vias públicas e a redução da poluição do ar pela redução da emissão de gás carbônico pelos veículos que realizam a coleta.

Determinar rotas ótimas para veículos coletores de lixo garantem a redução dos trajetos, o que consequentemente contribuem para a economia de combustível dos caminhões, tempo de trabalho dos funcionários, desgaste de peças e pneus, utilização de óleo de motor e, dependendo da situação redução da frota de veículos.

Visto que novas ruas são formadas e modificadas frequentemente, então sempre haverá necessidade de expandir suas rotas. Então, determinar o percurso dos veículos pode ser uma tarefa mais repetitiva do que parece e usar o conhecimento empírico ou basear-se na experiência de profissionais, não garantem boas rotas. Assim, o uso de um algoritmo baseado no PCCC pode ser de grande interesse e contribuição para estas empresas, reduzindo custos e tempo para se determinar as rotas.

Conforme abordado na literatura, o PCCC é um problema *NP-difícil*. Então, mesmo que existam diversos modelos que resolvam este problema de forma ótima, ainda assim, sua resolução é difícil e exaustiva computacionalmente. Para isso, abordam-se várias heurísticas, que mesmo não garantindo a otimalidade da solução, geralmente encontram soluções melhores do que as empíricas e com rápida resolução para a tomada de decisão.

A abordagem utilizada neste trabalho considera traçar uma única rota e, partindo desta, quebrá-la em rotas menores e coletar a demanda de cada arco exatamente uma vez, evitando extrapolar a capacidade do veículo coletor; e no final, determinar a rota para cada veículo. Essa abordagem é conhecida na literatura pelas heurísticas do tipo *Route-First Cluster-Second*. Abordagens do *Route-First Cluster-Second* para Problemas de Roteamento em Arcos podem ser encontradas em: Ulosoy (1985), Amberg, Domschke e Voβ (2000), Willemse e Joubert (2016), Van Bevern, Komusiewicz e Sorge (2017), Lenis e Rivera (2018). Tais trabalhos têm seus métodos descritos no Capítulo 3 deste trabalho. Entre as abordagens, não se encontrou na literatura um procedimento, onde na fase de divisão da rota única (*Cluster-Second*), utiliza-se a quebra de rotas por meio de mudanças de transição nos nós do grafo.

A contribuição desse trabalho parte da elaboração de um algoritmo heurístico, que buscará viabilizar rotas melhoradas, como suporte a coleta de lixo municipal, como também apresentar um programa desenvolvido em linguagem de programação *Python* que retornará uma rota para cada veículo da forma mais prática e rápida, economizando tempo e dinheiro para as empresas afins.

1.3 OBJETIVOS GERAL E ESPECÍFICOS

Nesta seção são apresentados o objetivo geral desta dissertação, assim como os objetivos específicos descrevendo os passos executados no escopo da pesquisa.

1.3.1 Objetivo geral

Propor um modelo de resolução do Problema do Carteiro Chinês Capacitado por meio de um procedimento heurístico para determinar rotas de coleta de lixo urbano viáveis considerando a capacidade dos coletores e comparar com um modelo ótimo presente na literatura.

1.3.2 Objetivos específicos

Os objetivos específicos da pesquisa são:

a) explorar o modelo de otimização para o Problema do Carteiro Chinês Capacitado (PCCC) implementado por Golden e Wong (1981);
b) desenvolver um modelo heurístico para resolução do PCCC como forma de determinar o percurso para os veículos coletores de lixo;
c) implementar o algoritmo heurístico por meio de uma linguagem de programação;
d) avaliar a heurística em dois casos reais na cidade de Recife-PE e comparar com um modelo matemático exato encontrado na literatura;
e) avaliar instâncias quanto a qualidade das soluções e tempos computacionais comparado ao modelo matemático exato encontrado na literatura.

2 REFERENCIAL TEÓRICO

No presente capítulo serão apresentados os principais conceitos e notações sobre grafos e suas principais propriedades pertinentes ao entendimento sobre Grafos Eulerianos, Roteamento em Arcos, o Problema do Carteiro Chinês e o Problema do Carteiro Chinês Capacitado.

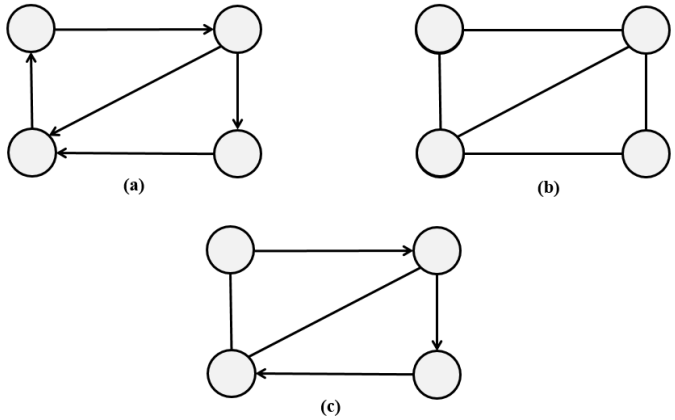
2.1 GRAFOS

Um grafo é representado por $G = (N, E \cup A)$, onde N é o conjunto finito de vértices (nós) de *G*, *E* representa o conjunto de arestas que formam os pares ordenados não direcionados de diferentes vértices e A representa o conjunto de arcos que formam os pares ordenados direcionados de diferentes vértices. Uma aresta é formada por um par ordenado $(i, j) \in E$, sendo *i* e *j* vértices do conjunto *N* que indicam uma ligação não direcionada entre eles. De forma análoga, um arco é formado por um par ordenado $(i, j) \in A$, onde os vértices *i* e *j* pertencem ao conjunto N e indicam uma ligação direcionada do vértice *i* para *j*.

O par (i, j) deve ser distinguido do par (j, i). Se (i, j) é um arco, então (i, j) representa uma saída do nó inicial *i* e uma entrada do nó final *j*. Quando a aresta une um par de nós *i* e *j* como (i, j) ou (j, i), então ele é não direcionado, uma vez que se pode percorrer os nós em ambas as direções. Quando (i, j) difere de (j, i), então ele possui uma única direção e é chamado arco direcionado, ou somente arco (AHUJA; MAGNANTI; ORLIN, 1993). Se *i* e *j* se ligam, então eles são adjacentes (TUCKER, 2012).

Um grafo dirigido ou digrafo é um grafo direcionado cujos arcos também são direcionados, um grafo não dirigido possui todas as arestas não direcionados, enquanto quando existem arcos direcionados e não direcionados no mesmo grafo, este é chamado de grafo misto (AHUJA; MAGNANTI; ORLIN, 1993). A Figura 1 expõe grafos que representam os tipos: direcionado (a), não direcionado (b) e misto (c).

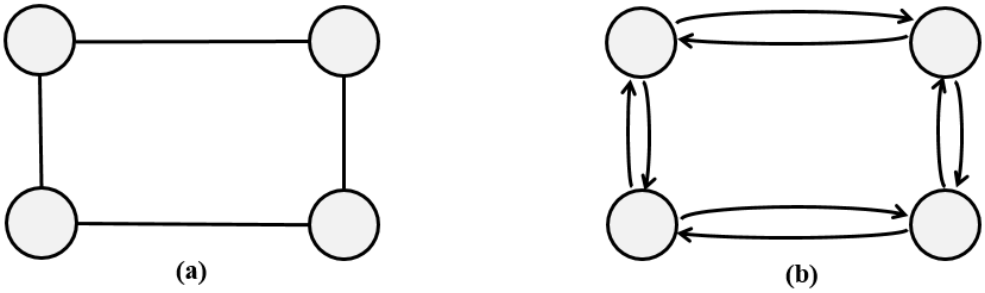
Figura 1 – Grafo direcionado (a), não direcionado (b) e misto (c)



Fonte: O autor (2020)

Bertsekas (1998) expõe que em muitos contextos e casos práticos se torna mais adequado usar grafos direcionados para representar grafos não direcionados. Para tanto, basta substituir o arco não direcionado (i, j) por dois arcos direcionados (i, j) e (j, i). A Figura 2 apresenta esta transformação do grafo não dirigido (a) para o dirigido (b).

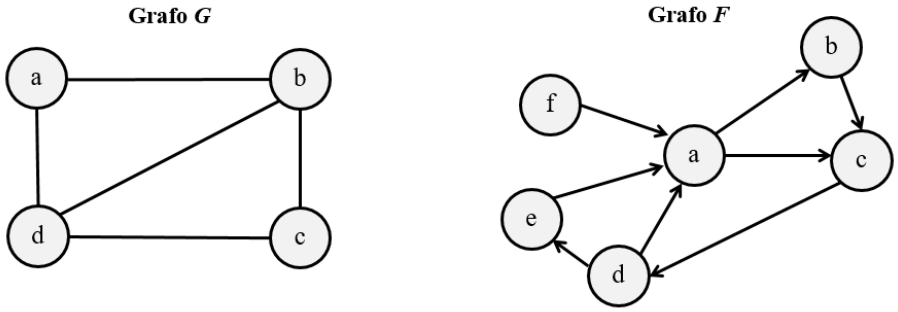
Figura 2 – Transformando um grafo não dirigido (a) em dirigido (b)



Fonte: O autor (2020)

O grau de um vértice *i* em um grafo *G* é denotado por $\deg_G(i)$, e é o número de arestas incidentes a ele. Em um grafo dirigido, o grau de um vértice qualquer é a soma dos arcos que entram neste vértice (*indegree*) e dos arcos que estão saindo deste vértice para os demais (*outdegree*) (BERTSEKAS, 1998; KELLER; TROTTER, 2015). A Figura 3 mostra um grafo simples *G* cujos graus dos vértices são: $\deg_G(b) = \deg_G(d) = 3$, $\deg_G(a) = \deg_G(c) = 2$ e um grafo *F* que tem grau $\deg_G(a) = \deg_{in}(a) + \deg_{out}(a) = 3 + 2 = 5$.

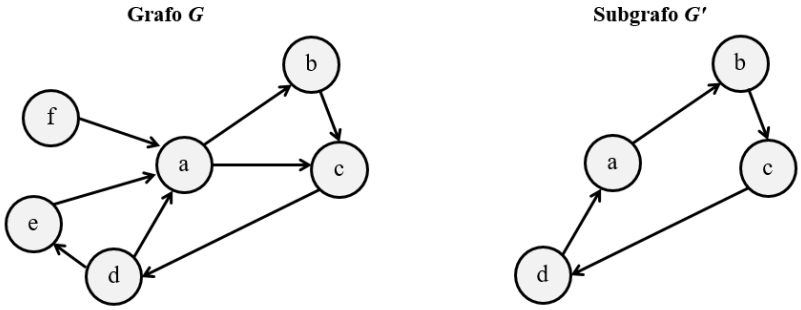
Figura 3 – Grafo G e Grafo F



Fonte: O autor (2020)

Um subgrafo G' do grafo G é um grafo formado por um subconjunto de vértices e arestas do grafo original (TUCKER, 2012). $G' = (N', E')$ é um subgrafo de um grafo $G = (N, E)$, se $N' \subseteq N$ e $E' \subseteq E$. Se H é um subgrafo de G, então G contém H. Se $G' \neq G$, então G' é chamado um subgrafo próprio de G. Se $N' = N$, então G' é um subgrafo de extensão de G (SCHRIVER, 2003). Um grafo G e um subgrafo G' de G são apresentados na Figura 4.

Figura 4 – Grafo G e seu subgrafo G'



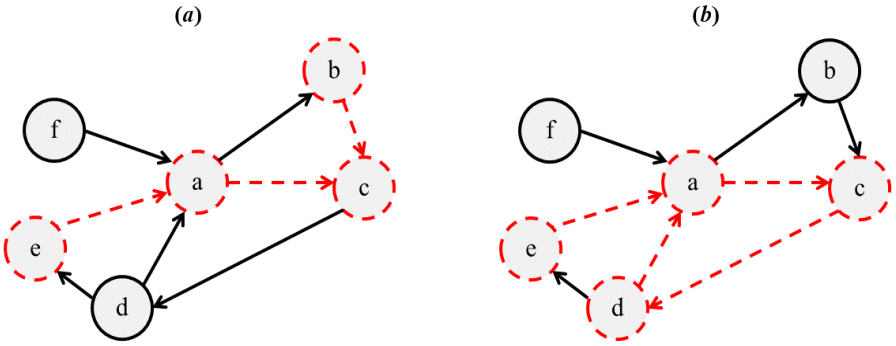
Fonte: O autor (2020)

Após a definição dos conceitos básicos acerca de grafo, precisa-se definir algumas noções sobre suas propriedades para compreensão dos termos abordados mais à frente.

Um passeio em um grafo direcionado $G = (N, A)$ é um subgrafo de G cuja sequência de vértices e arcos, é dado sendo $(i_1, a_1, i_2, a_2, \dots, i_{r-1}, a_{r-1}, i_r)$, onde r é o número de vértices presentes no passeio, conforme a propriedade que $a_k = (i_k,\ i_{k+1}) \in A$ ou $a_k = (i_{k+1},\ i_k) \in A$ para todo $1 \leq k \leq r-1$ (AHUJA; MAGNANTI; ORLIN, 1993). Os vértices i_1 e i_r são chamados de vértices inicial e final, respectivamente. Similar, a_1 e a_{r-1} são chamados de arcos inicial e final, respectivamente (SCHRIJVER, 2003). Já o passeio dirigido (ou direcionado) se trata de um passeio em um grafo respeitando o sentido dos arcos, ou seja, $a_k = (i_k, i_{k+1}) \in A$

deverá ser satisfeita. (AHUJA; MAGNANTI; ORLIN, 1993). A Figura 5 apresenta um passeio (a) e um passeio dirigido (b) de um grafo nas linhas tracejadas.

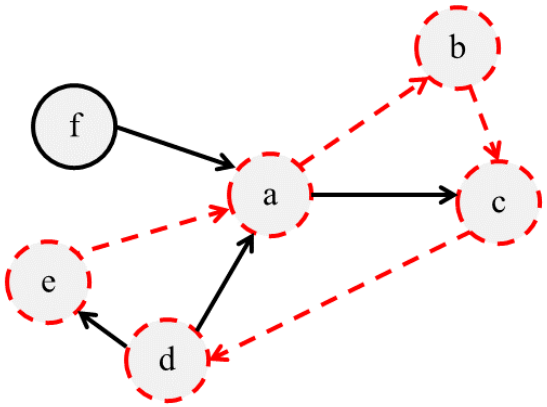
Figura 5 – Passeio (a) e passeio dirigido (b) de um grafo



Fonte: O autor (2020)

Quando todos os vértices incluídos no passeio (passeio dirigido) são distintos, então a sequência é chamada de caminho (caminho dirigido). Para enfatizar onde um caminho começa e termina, uma sequência de vértices distintos, tal que, $(i_1, i_2, \dots, i_{k-1}, i_k)$ representa um caminho de i_1 para i_k em G com k vértices e $k-1$ arcos. Os nós i_1 e i_k são chamados de nó inicial (ou origem) e nó final (ou destino) de G, respectivamente (BERTSEKAS, 1998; KELLER; TROTTER, 2015). O grafo da Figura 6 demonstra um caminho no grafo nas linhas tracejadas.

Figura 6 – Caminho em um grafo

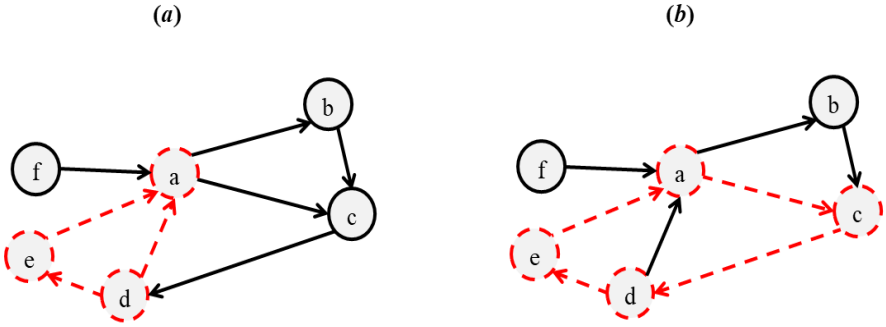


Fonte: O autor (2020).

Quando o caminho em G $(i_1, i_2, \dots, i_{k-1}, i_k)$ apresentar o vértice inicial igual ao final $(i_1 = i_k)$, independente da sequência dos arcos, recebe o nome de ciclo (BERTSEKAS, 1998;

SCHRIJVER, 2003). O ciclo dirigido é um caminho dirigido $(i_1, i_2, \dots, i_{k-1}, i_k)$ junto com o arco direcionado (i_k, i_1) (AHUJA; MAGNANTI; ORLIN, 1993). Já o circuito de um grafo é uma sequência de vértices $(i_1, i_2, \dots, i_{k-1}, i_k)$ onde $i_1 = i_k$ e todo i_n é adjacente a um i_{n+1} (TUCKER, 2012), portanto é um caminho fechado. O grafo da Figura 7 apresenta um ciclo (a) e um circuito (b).

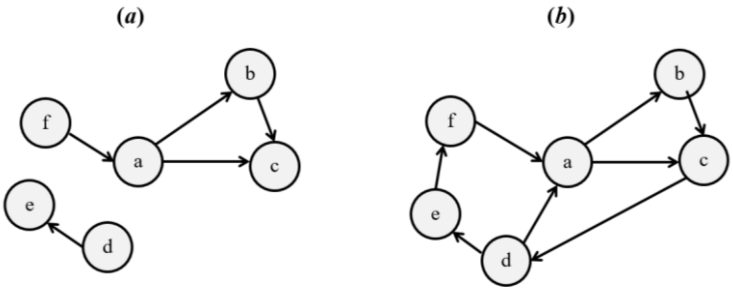
Figura 7 – Ciclo (a) e circuito (b) de um grafo



Fonte: O autor (2020).

Um grafo $G = (N, E)$ é dito conexo quando houver para quaisquer dois vértices *u* e *v* um caminho conectando *u* e *v*. Um grafo conexo *G* se torna desconexo quando algumas arestas ou vértices são removidas, não existindo caminhos que unam pelo menos um par de vértices (TUCKER, 2012). Um grafo conexo é fortemente conexo se, e somente se, nele existir pelo menos um caminho direcionado que ligue cada nó a cada outro nó (AHUJA; MAGNANTI; ORLIN, 1993). A Figura 8 mostra um grafo desconexo (a) e um fortemente conexo (b). Note que em (b) existem caminhos direcionados ligando o nó *c* para qualquer outro nó.

Figura 8 – Grafo desconexo (a) e grafo fortemente conexo (b)



Fonte: O autor (2020).

Um ciclo que percorre todos os nós de um grafo, exatamente uma vez, recebe o nome de ciclo hamiltoniano, independente da sequência e direção dos arcos. Quando o caminho for

um circuito que percorre todos os nós, então recebe o nome de circuito hamiltoniano. Um grafo *G* é hamiltoniano se há pelo menos um circuito hamiltoniano (SCHRIJVER, 2003).

Um ciclo que percorre todos os arcos/arestas de um grafo, exatamente uma vez, recebe o nome de ciclo Euleriano (ou de Euler). De forma análoga, se um grafo *G* apresentar um circuito em que cada um de seus arcos/arestas é percorrido apenas uma vez, este é um grafo Euleriano (BERTSEKAS, 1998).

Segundo Larson e Odoni (1981) o emprego de grafos é amplamente aplicado em diversas áreas de conhecimento, dando ênfase aos problemas de redes urbanas. Assim, partindo dos conceitos fixados pode-se explorar tais problemas em redes urbanas que estão generalizados nos problemas de roteamento.

2.2 PROBLEMA DE ROTEAMENTO

Os problemas de roteamento são problemas de otimização combinatória que buscam encontrar rotas (percursos) em um grafo específico (SHERAFAT, 2013). Eles podem ser classificados em dois grupos distintos. Os problemas de roteamento em nós buscam determinar uma ou mais rotas através das quais um veículo percorre alguns ou todos os nós de um grafo e os problemas de roteamento em arcos objetivam encontrar uma rota que passa por alguns ou todos os arcos de um grafo (BATISTA; SCARPIN, 2015).

Segundo Larson e Odoni (1981), os problemas de roteamento podem ser divididos em diversos outros problemas, como seguem abaixo.

a) Problema do Caixeiro Viajante (PCV): requer a determinação de um circuito de custo mínimo que percorre todos os nós de um dado grafo;
b) Problema do Carteiro Chinês (PCC): consiste em determinar um circuito de custo mínimo que contém todos os arcos e/ou arestas do grafo, pelo menos uma vez;
c) Problema do Carteiro Chinês Com Vento (PCCV): tem a mesma formulação de PCC num grafo não dirigido, exceto que o custo de percurso de cada aresta varia a depender em que sentido a mesma é percorrida (como se tivesse vento a favor, ou vento contra);
d) Problema do Carteiro Rural (PCR): consiste em determinar um circuito de custo mínimo que contém um subconjunto de arcos (requeridos) do grafo. É o caso que se aproxima mais a problemas reais de distribuição;

Para variar ainda mais, todos os casos acima podem ser formulados em grafos não dirigidos, dirigidos, ou mistos, o que em geral implica no método de solução. Com exceção de

PCC formulados em grafo dirigido e não dirigido, todos os demais casos são problemas *NP-difícil* (DROR, 2012).

2.2.1 Problemas de Roteamento em Arcos

A década de 1970 foi marcada por grande avanço no desenvolvimento de modelos matemáticos destinados à resolução dos Problemas de Roteamento em Arcos (PRA) (DOS SANTOS, 2016). Exemplos desse tipo de problema, podem ser vistos na limpeza e varrição de ruas, na lavoura de neve após uma tempestade de neve, na medição de leituristas, na entrega de correspondências em residências e na coleta de lixo residencial (LARSON; ODONI, 1981).

Problemas de Roteamento em Arcos aparecem em diversas áreas de logística e pesquisa operacional. Conforme Eiselt, Gendreau e Laporte (1981), o objetivo dos problemas de roteamento em arcos consiste em determinar a(s) rota(s) de menor custo para visitar alguns arcos do grafo, com uma ou mais restrições complementares.

Estima-se que bilhões de dólares são despendidos em serviços de roteamento anualmente nos Estados Unidos, principalmente através de recursos públicos, criando-se assim uma demanda por métodos que otimizem os investimentos na área. Nas últimas décadas, o avanço no poder de processamento dos computadores e novas técnicas de otimização contribuíram para a disseminação e adoção de sistemas de software para roteamento em arcos (DROR, 2012).

Ao longo dos anos foram surgindo diversos modelos de PRA’s cada vez mais complexos e visando aplicações mais práticas. Dentre os mais estudados pode-se citar o Problema do Carteiro Chinês (PCC), o Problema do Carteiro Rural (PCR) e o Problema de Roteamento em Arcos Capacitado (PRAC).

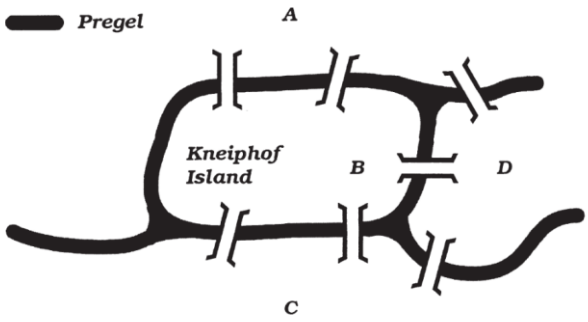
O primeiro trabalho documentado sobre roteamento em arcos foi estudado há quase três séculos, o conhecido problema das sete pontes de Königsberg, que deu início a discussão sobre a teoria dos grafos e sobre grafos Eulerianos.

2.3 GRAFOS EULERIANOS

A teoria dos grafos teve suas origens há quase três séculos em um problema na cidade de Königsberg na Prússia, atualmente a cidade de Kaliningrad na Rússia. O rio Pregel que atravessa a cidade tem duas grandes ilhas, que foram conectadas à cidade por sete pontes (Figura 9). Perguntando-se se era possível percorrer a cidade, de tal forma, que se pudesse

atravessar as sete pontes cada uma exatamente uma vez, o matemático Leonhard Euler, em 1736 solucionou este problema aplicando a teoria dos grafos (KELLER; TROTTER, 2015).

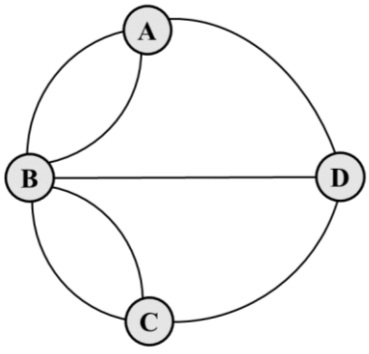
Figura 9 – As sete pontes de Königsberg



Fonte: Dror (2012)

Euler observou que o problema das pontes de Königsberg poderia ser representado por um grafo não direcionado (Figura 10) no qual cada aresta corresponde as sete pontes e os vértices exprimem cada uma das duas margens do rio das duas ilhas (DROR, 2012). Euler provou que a questão seria análoga a encontrar um caminho no grafo que envolvesse cada aresta exatamente uma vez, o que passou a ser chamado de caminho de Euler (caminho Euleriano) (ASSAD; GOLDEN, 1995).

Figura 10 – Grafo representando as pontes de Königsberg



Fonte: Adaptado de Dror (2012).

Para o problema das sete pontes de Königsberg, visto que um caminho fechado requer que cada vértice tenha o mesmo número de *indegrees* e *outdegrees*, obviamente o caso não teve solução devido todos os vértices terem grau ímpar (DROR, 2012). Porém, os grafos com a característica de apresentarem passeios fechados que percorrem todas as arestas exatamente

uma vez recebem o nome de grafo balanceado ou Euleriano (DROR, 2012). Isso é uma definição inicial para se estabelecer uma trilha e um *tour* de Euler.

Quando existir uma sequência de vértices não obrigatoriamente distintos $(x_1, x_2, \dots, x_n)$ cujos vértices consecutivos são unidos por uma aresta, em que nenhuma aresta poderá ser repetida, tem-se uma trilha *T*. Uma trilha de Euler é uma trilha que contém todas as arestas em um grafo (e visita cada vértice pelo menos uma vez) (TUCKER, 2012).

Um *tour* é uma trilha fechada em um grafo percorrendo todas as arestas de *G*, no mínimo uma vez. Em um grafo *G*, uma sequência alternada de nós e arestas $(n_1, e_1, n_2, e_2, n_3, \dots, n_l, e_l, n_{l+1})$, de tal forma que a aresta e_i encontre os nós (distintos) n_i e n_{i+1} e que o nó inicial n_1 seja igual ao nó final n_{l+1}, esta sequência é um *tour* (EDMONDS, JOHNSON, 1973). Um *tour* recebe o nome de *tour* de Euler, se, e somente se, este incluir cada aresta de *G* exatamente uma vez (ASSAD; GOLDEN, 1995). Um grafo *G* é chamado de grafo Euleriano (ou grafo de Euler) se contiver um *tour* de Euler em *G* (AHUJA; MAGNANTI; ORLIN, 1993).

Euler, embora não tenha resolvido o problema das sete pontes na época, deu as condições necessárias para a existência de um grafo Euleriano. Assim, para se determinar um circuito Euleriano em um grafo *G* deve-se apresentar as características de conectividade e o grau dos nós deste grafo. Assim, para que um circuito (ciclo) de Euler exista em um grafo, todos os vértices devem ter grau par (Teorema 1). Para o caso de um grafo apresentar vértices de grau ímpar, então não haverá um circuito de Euler a menos que algumas arestas sejam percorridas mais de uma vez (ASSAD; GOLDEN, 1995).

O teorema 1 foi proposto por Euler em 1736 para definir a existência de um grafo Euleriano.

Teorema 1 *- Um grafo G, fortemente conexo, contém um circuito Euleriano, se, e somente se, o grafo tem apenas nós de grau par.*

Prova: O entendimento do Teorema 1 sobre o grafo ser fortemente conexo refere-se a característica de que qualquer circuito Euleriano percorre todos os nós e arestas do grafo partindo de um nó inicial qualquer (conectividade). Pelo fato de todo circuito Euleriano apresentar cada aresta sendo percorrida exatamente uma única vez, então sempre haverá uma aresta entrando em um nó *i* e uma aresta saindo de *i*, então a incidência de arestas nos nós forma um conjunto de números pares. Em grafos direcionados, o *indegree* do nó deve ser igual ao seu *outdegree*. A soma dessas parcelas é um número par de arestas chegando e saindo em cada nó do grafo. ■

A construção de um grafo Euleriano é apresentada de forma simples por Sherafat (2004).

Para estabelecer um circuito Euleriano em um grafo *G* fortemente conexo e com nós de grau par inicia o percurso em um nó qualquer n_i, cobre uma aresta adjacente que ainda não foi utilizada para chegar a um outro nó n_j, e repete o procedimento cobrindo as arestas não visitadas partindo para demais nós. Cada vez que se incide em um nó deverá existir uma aresta que ainda não foi percorrida, devido ao grau de todos os nós serem par. Por fim, o procedimento terminaria exatamente no mesmo nó de partida n_i, completando um circuito *C*.

Se *C* apresentar todas as arestas de *G*, então chega-se a um circuito Euleriano; caso contrário, deve-se identificar o grafo parcial *G'*, formado pelas arestas de *G* que ainda não foram utilizadas. Da mesma forma, todos os nós de *G'* também são de grau par. Deve-se, então, identificar um nó n_j contido em *C* e ligado a um nó n_k não contido em *C* por meio de uma aresta ainda não coberta de *G'*. A ligação (n_j, n_k) existe, visto que *G* é fortemente conexo. A partir de n_k deve-se cobrir as arestas que não estão em *C* formando um novo circuito *C'*, o qual será agrupado a *C*, a partir de n_j, formando um único circuito. Tal procedimento é repetido até que se chegue a um circuito Euleriano completo no grafo *G*.

Após a determinação do grafo ser Euleriano, determinar um circuito Euleriano para este é considerado trivial. Segundo Sherafat (2004), uma importante particularidade de grafos não Eulerianos é de possuir um número par de nós de grau ímpar.

Teorema 2 *- Todo grafo possui um número par de nós de grau ímpar.*

A explicação é devido à consideração da quantidade total de incidências que os nós de um grafo possuem. Dado *m* ser o número de arestas de um grafo e admitindo que cada aresta liga dois nós, logo a soma de graus de todos os nós do grafo é *2m*, que é um número par. Para a soma de todos os graus dos nós pares: g_p e g_i a soma de todos os graus dos nós ímpares. Tem-se $g_p + g_i = 2m$. Logicamente, g_p deve ser um número par, assim g_i torna-se também um número par, e se a soma de alguns números ímpares é par, então é em razão da quantidade dos números ímpares ser par. Por fim, pode-se afirmar que em qualquer grafo há um número par de nós de grau ímpar.

Como a atividade principal do problema das sete pontes de Königsberg era de percorrer certos arcos de uma rede de transporte, então este problema pode ser visto como a primeira aplicação de roteamento de arco (ASSAD; GOLDEN, 1995). Entre os Problemas de Roteamento em Arcos, o Problema do Carteiro Chinês (PCC) está entre os mais famosos e mais abordados na literatura (SHAFAHI; HAGHNI, 2015).

2.4 PROBLEMA DO CARTEIRO CHINÊS

Nesta seção, é abordado diversos aspectos do Problema do Carteiro Chinês desde a sua contextualização histórica, as principais definições e as suas variações para diferentes tipos de grafos.

2.4.1 Contexto histórico

O Problema do Carteiro Chinês (PCC) é um problema de otimização que pertence à classe de problemas de roteamento em arcos, cujo objetivo é encontrar um *tour* em um grafo considerando a menor distância ou custo (DROR, 2012). Para Goldbarg e Luna (2005) o *tour* do carteiro distingue-se do circuito Euleriano porque nele é permitida, quando necessário, a repetição de arestas.

O PCC foi introduzido pela primeira vez por Meigu Guan (ou Mei-Ko Kwan na antiga transliteração) (ASSAD; GOLDEN, 1995), em um artigo chinês, onde Guan introduziu o problema de determinar a rota mais curta de um carteiro (SCHRIJVER, 2003). Guan foi um matemático chinês, que durante a Revolução Cultural Chinesa permaneceu durante algum tempo como funcionário dos correios (EISELT; GENDREAU; LAPORTE, 1995).

Em contraste com o problema das pontes de Königsberg, que se preocupa apenas com a existência e determinação de um caminho fechado, Guan passou a trabalhar com os grafos não Eulerianos (EISELT; GENDREAU; LAPORTE, 1995), ou seja, para determinar uma rota de distância mínima cobrindo cada aresta (as ruas de uma cidade) pelo menos uma vez.

Enquanto Guan (1962) procurava desenhar uma rota para um carteiro, acabou descobrindo o seguinte problema: um carteiro deverá cobrir a sua rota na qual foi designado antes de retornar aos correios percorrendo a menor distância a pé. Segundo Assad e Golden (1995) o problema proposto por Guan envolvia adicionar um conjunto de arestas ao grafo para garantir que todos os nós do grafo tenham grau par. Ainda, segundos os autores, a solução proposta por Guan não produzia um algoritmo polinomial para o problema do carteiro.

Edmonds (1965) entendeu que o problema de Guan em adicionar um conjunto de arestas de custo mínimo para obter um grafo com todos os nós de grau par poderia ser resolvido como um "problema de emparelhamento (*matching*)" de forma eficiente. Edmonds e Johnson (1973) desenvolveram algoritmos para grafos não direcionados, direcionados e mistos, combinando dois algoritmos conhecidos até então: o algoritmo de caminho mais curto e um algoritmo para o emparelhamento máximo (SCHRIJVER, 2003).

Mesmo Guan não tendo chegado a solução exata, mas pela introdução do tema e a publicação que motivou os problemas de otimização de rede, o problema ficou conhecido como o Problema do Carteiro Chinês, em homenagem ao matemático chinês (ASSAD; GOLDEN, 1995). Além disso, o artigo de Guan se mostrou importante, porque abordou pela primeira vez o "problema de grafos aumentados", problema que consiste em determinar um grafo Euleriano, da maneira mais econômica, introduzindo arestas extras (DROR, 2012). O problema de aumento é, de fato, o problema central do roteamento em arcos. Uma vez obtido um grafo balanceado, determinar um percurso real é relativamente fácil.

2.4.2 Definição

Por definição, o PCC padrão é um problema no qual, dado um grafo $G = (N, E)$ cujas arestas (i, j) possuem um comprimento não negativo $c_{ij} > 0$, deseja-se identificar um caminho de comprimento mínimo que se inicie em algum nó, passe por todas as arestas de G pelo menos uma vez e retorne ao nó inicial (AHUJA; MAGNANTI; ORLIN, 1993).

O PCC pode ser aplicado em grafos não direcionados, dirigidos e mistos. Para cada caso, existem requisitos necessários para garantir que um grafo seja balanceado e, portanto, que seja determinado o *tour* do carteiro. Eiselt, Gendreau e Laporte (1995) apresentam essas condições:

a) grafo não direcionado: o grafo deve ser conexo e apresentar todos os nós de grau par, ou seja, ter as condições propostas por Guan (1962);

b) grafos dirigidos: o grafo precisa ser fortemente conexo e todos os nós devem ter grau par, ou seja, o número de arcos incidentes a um nó deve ser igual ao número de arcos saindo do mesmo nó (*indegree* igual ao *outdegree*);

c) grafos mistos: grafo fortemente conexo e todos os nós precisam ter grau par para ambas arestas e arcos.

O PCC em grafos não dirigidos e dirigidos possuem algoritmos e modelos matemáticos que apresentam soluções exatas em tempo polinomial, porém em grafos mistos, o problema é considerado *NP*-completo quanto a sua complexidade (WANG; WEN, 2002).

Será abordado cada caso do PCC de forma separada apresentando os algoritmos e modelos matemáticos para resolução como problema de programação linear, presentes na literatura.

2.4.3 Problema do Carteiro Chinês Não Direcionado (PCCND)

Seja $G = (N, E)$ um grafo não direcionado cujas arestas tenham pesos conhecidos para todo $(i, j) \in E$. Caso G seja conexo e todos os seus nós tenham grau par, então o PCCND pode ser resolvido encontrando um circuito que atravesse cada aresta de G uma única vez e para o qual a soma dos pesos seja a mínima possível (LARSON; ODONI, 1981), ou seja, o grafo precisa ser balanceado (Euleriano). Para tanto, a aplicação do PCCND é oriunda diretamente dos trabalhos de Euler (1736) e Guan (1962) e sua resolução pode ser feita por qualquer algoritmo eficiente que encontre um circuito Euleriano.

Godinho Filho e Junqueira (2006) apontaram diversos algoritmos que encontram um circuito Euleriano em um grafo não direcionado, entre eles, os autores citam o algoritmo de Fleury (Kauffman, 1967), o algoritmo de Edmonds e Johnson (1973), o algoritmo de Larson e Odoni (1981) apresentado abaixo, entre outros.

Algoritmo 1. *Obtenção de um tour de Euler em um grafo Euleriano*.

Dado um grafo $G = (N, E)$ conexo e com todos os nós de grau par, então inicie o *tour* em um nó $n_0 \in N$ qualquer desejado e atravesse sucessivamente as arestas adjacentes em G sempre excluindo cada uma aresta percorrida. Durante o procedimento não percorra uma aresta que é uma ponte, ou seja, uma aresta cuja ao ser apagada torna o grafo desconexo. O processo deve continuar até que todas as arestas de G tenham sido excluídas e que se tenha retornado ao nó inicial n_0, tornando a rota percorrida, um *tour* de Euler.

Quando o grafo não é Euleriano, ou seja, possui nós de grau ímpar, é necessário torná-lo balanceado a partir da duplicação de arestas existentes em alguns nós. Para Larson e Odoni (1981) isso deve ser feito selecionando a combinação das arestas duplicadas que resulte no *tour* de comprimento mínimo no final. Os autores explicam que duplicar uma aresta já existente no grafo original significa, basicamente, que aquela aresta deverá ser percorrida duas vezes no *tour* do carteiro chinês.

Um modelo matemático capaz de resolver o PCCND foi estabelecido por Edmonds e Johnson (1973) como um problema de programação linear inteira de minimização. A formulação abaixo está presente em Goldbarg e Luna (2005).

$$Minimize \sum_{(i,j) \in E} c_{ij} x_{ij} \tag{2.1}$$

$sujeito\ a$:

$$\sum_{(i,j) \in E} x_{ij} - \sum_{(j,i) \in E} x_{ji} = 0\,, \quad \forall\, i \in N \tag{2.2}$$

$$x_{ij} + x_{ji} \geq 1, \quad \forall\, (i,j) \in E \tag{2.3}$$

$$x_{ij} \text{ é } inteiro, \quad \forall\, (i,j) \in E \tag{2.4}$$

A variável x_{ij} representa o número de vezes que a aresta (i,j) deverá ser percorrida do nó *i* para o nó *j* durante o *tour*. O peso da aresta (i,j) é representado por c_{ij}, que na prática pode representar custo, distância, consumo de combustível, etc. A função objetivo (2.1) minimiza o custo total do *tour* do carteiro. As restrições (2.2) asseguram a conservação do fluxo, ou seja, que o número de arestas que entram no nó *i* será igual ao número de arestas que irá deixá-lo; (2.3) garantem que nenhuma aresta deixará de ser percorrida, seja de *i* para *j* ou vice-versa; e (2.4) indica que x_{ij} pertence ao conjunto dos números inteiros.

2.4.4 Problema do Carteiro Chinês Direcionado (PCCD)

Para o caso direcionado, tem-se um grafo direcionado $G = (N, A)$ que possui arcos com pesos não negativos c_{ij} para todo $(i,j) \in A$. Neste caso, o grafo não basta apenas ser conexo, como no caso não direcionado. É necessário a garantia de que G seja fortemente conexo (EISELT; GENDREAU; LAPORTE, 1995; WANG; WEN, 2002). Outra condição necessária para a existência de um *tour* de Euler em um grafo direcionado é que este deve ser simétrico, ou seja, para cada nó do grafo o número de arcos que entram deve ser igual ao número de arcos saindo (ASSAD; GOLDEN, 1995).

Portanto, quando as duas condições necessárias são satisfeitas, encontrar um *tour* de Euler pode ser facilmente resolvido por qualquer algoritmo que encontre um circuito Euleriano em um grafo não direcionado adaptado para grafos direcionados. Como é o caso do algoritmo de Fleury presente em Christofides (1975) e do algoritmo de Hierholzer presente em Hierholzer e Wiener (1873).

O algoritmo de Hierholzer é uma homenagem ao matemático alemão Carl Hierholzer, que infelizmente morreu muito jovem para ver seu trabalho ser publicado. Porém, sua contribuição foi reconhecida através da nomeação de um procedimento eficiente para determinação de circuitos Eulerianos (SAOUB, 2017). O algoritmo de Hierholzer tem como principal característica percorrer um conjunto de circuitos fechados distintos e mesclá-los em um único circuito por meio de nós afins (STERN; GERTSBAKH, 2019).

O algoritmo de Hierholzer inicia encontrando um circuito arbitrário originado de um nó inicial. Se este circuito contiver todas os arcos do grafo, um circuito Euleriano foi encontrado; caso contrário, outro circuito é acrescentado ao já existente (SAOUB, 2017), porém este só determina um circuito único se o grafo for Euleriano. Um *tour* de Euler pode ser encontrado em tempo linear usando o algoritmo de Hierholzer (ALSTRUP et al., 2018), como implica o algoritmo 2.

Algoritmo 2: *Algoritmo de Hierholzer para grafos direcionados*

1. Inicie percorrendo qualquer arco do nó inicial e selecione os arcos vizinhos ainda não visitados até que se forme um circuito;

2. Se ainda houver arcos não visitados, comece de um nó que pertença ao circuito já existente, mas que tenha um arco não visitado e crie um novo circuito como na primeira etapa;

3. Se todos os arcos foram visitados, então um circuito Euleriano deve ser construído a partir dos circuitos existentes, juntando-os a partir de um nó comum.

O algoritmo 2, como já mencionado, serve apenas para grafos Eulerianos. Quando o grafo não é Euleriano, precisa-se torná-lo Euleriano, onde a ideia é análoga ao caso não direcionado. Conforme Assad e Golden (1995) cópias adicionais de alguns arcos precisam ser incluídas para realizar o *tour*. Tal procedimento pode ser feito atribuindo a x_{ij} o número de vezes que o arco (i,j) é percorrido. A escolha ideal de x_{ij} pode ser determinada considerando o caso como um problema de fluxo de custo mínimo, onde o resultado estabelece o menor custo (comprimento) total do *tour* do carteiro (AHUJA; MAGNANTI; ORLIN, 1993; ASSAD; GOLDEN, 1995).

Wang e Wen (2002) utilizaram o modelo matemático (2.5) – (2.7) para resolução exata do PCCD por meio de Programação Linear.

$$Minimize \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{2.5}$$

$sujeito\ a$:

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(i,j)\in A} x_{ji} = 0, \ \ \forall\, i \in N \tag{2.6}$$

$$x_{ij} \geq 1\ e\ inteiro\ , \quad \forall (i,j) \in A \tag{2.7}$$

No modelo matemático, a função objetivo (2.5) minimiza o custo total do carteiro em seu percurso direcionado. Quanto as restrições, (2.6) garantem a conservação de fluxo, cada nó tem seu *indegree* igual ao seu *outdegree* e; (2.7) indicam que cada arco (i,j) seja percorrido pelo menos uma vez em G.

Uma vantagem deste modelo matemático é que sua matriz de restrições, conforme abordado por Wang e Wen (2002), é totalmente unimodular, essa propriedade garante que a resolução dada pela programação inteira seja a mesma solução da programação linear no caso de a variável inteira ser relaxada, tornando o método de solução mais rápido, especialmente para redes de grande porte. A subseção 2.4.4.1 apresenta uma definição de unimodularidade.

O PCCD também pode ser formulado e resolvido em programação matemática como um problema clássico de transportes, como abordado por Beltrami e Bodin (1974) e Orloff (1974). Seja qual for a abordagem para determinar quais os arcos que sejam acrescentados, após tal procedimento, pode-se utilizar os diversos algoritmos existentes na literatura, como também, o algoritmo 2 aqui abordado.

2.4.4.1 Matriz Totalmente Unimodular

O objetivo desta subseção não é provar teoremas acerca de unimodularidade de matrizes, mas apenas introduzir de forma sucinta e breve a definição de matriz totalmente unimodular e sua relação com problemas de rede. Detalhes e provas de teoremas podem ser encontrados em Ahuja, Magnanti e Orlin (1993) e Schrijver (1998).

Uma matriz A não singular, ou seja, que pode ser inversa, recebe o nome de unimodular se A for uma matriz inteira quadrada (tamanho p x p) e apresentar determinante igual a +1 ou -1, ou seja, $det(A) = +1$ ou $det(A) = -1$.

Uma matriz A é totalmente unimodular se A é uma matriz p x q com elementos inteiros e p linhas linearmente independentes e se toda sub matriz quadrada B de A tiver determinante igual a 0, +1 ou -1. Em particular, cada entrada em uma matriz totalmente unimodular tem valores 0, +1 ou -1. Uma ligação entre totalmente unimodular e a programação linear inteira é fornecida em Ahuja, Magnanti e Orlin (1993) e em Schrijver (1998). O teorema a seguir mostra a relação entre a unimodularidade e solução inteira de programas lineares.

Teorema 3 *- Seja A uma matriz inteira com linhas linearmente independentes. Em seguida, as três condições a seguir são equivalentes.*

a) A é unimodular.
b) toda solução viável básica definida pelas restrições $Ax = b$, onde $x \geq 0$ é um número inteiro para qualquer vetor inteiro b.
c) toda sub matriz quadrada B de A tem uma inversa inteira B^{-1}.

O Teorema 3 apresenta uma prova algébrica da propriedade de integralidade dos fluxos de rede, presente em Ahuja, Magnanti e Orlin (1993), onde os modelos de fluxo de rede têm soluções ótimas inteiras, porque todas as matrizes de incidência de arco-nó são totalmente unimodulares para grafos dirigidos, como apresenta o Teorema 4 e sua prova.

Teorema 4 *- A matriz de incidência de arco-nó A de um grafo dirigido é totalmente unimodular.*

Prova: Para provar o teorema, precisa-se mostrar que cada sub matriz quadrada B de A de tamanho k tem determinante 0, + 1 ou -1. Para isso, se estabelece esse resultado realizando indução em k. Como cada elemento de A é 0, + 1 ou -1, então o teorema é verdadeiro para $k = 1$. Agora, suponha que o teorema seja válido para alguns k. Seja B qualquer sub matriz de $(k + 1)$ x $(k + 1)$ de A. A matriz B satisfaz exatamente uma das três seguintes possibilidades:

a) B contém uma coluna sem elemento diferente de zero;
b) toda coluna de B possui exatamente dois elementos diferentes de zero; nesse caso, um deles deve ser + 1 e o outro -1; e
c) alguma coluna B_l possui exatamente um elemento diferente de zero, para a i-ésima linha.

No caso (1), o determinante de B é 0 e o teorema é válido. No caso (2), a soma de todas as linhas em B produz o vetor 0, implicando que as linhas em B são linearmente dependentes e, consequentemente, $\det(B) = 0$. No caso (3), B' indica a sub matriz de B obtido pela exclusão da i-ésima linha e da l-ésima coluna. Então $\det(B) = \pm 1 \det(B')$. Pela hipótese de indução, $\det(B')$ é 0, + 1 ou -1, então $\det(B)$ também é 0, + 1 ou -1. Esta conclusão estabelece o teorema. ■

Por fim, as propriedades de unimodularidade fornecem um resultado muito forte, onde qualquer solução viável básica é garantida como valor inteiro sempre que o vetor do lado direito b for inteiro.

2.4.5 Problema do Carteiro Chinês Misto (PCCM)

Uma rede mista é representada por um grafo $G = (N, E \cup A)$ com um conjunto de arestas não orientadas E, um conjunto de arcos orientados A e pesos c_{ij} em todas as arestas e arcos. Como nos casos anteriores, o PCCM destina-se a encontrar um *tour* de mínimo custo para o carteiro em um grafo misto. Portanto, para garantir uma solução ao PCCM deve-se

estabelecer requisitos que assegurem G ser balanceado (ASSAD; GOLDEN, 1995; EISELT, GENDREAU; LAPORTE, 1995).

As condições necessárias para o grafo ser balanceado são: verificar se G é fortemente conexo e se em todos os nós de G incidem arcos e arestas de grau par, como mostrado por Eiselt, Gendreau e Laporte (1995): todo nó deve ser incidente com um número par de arcos e/ou arestas; ademais, se para todo subconjunto de nós (isto é, $S \subseteq V$), a diferença entre o número de arcos saindo do subconjunto S (S para $V \setminus S$) e o número de arcos entrando no subconjunto S ($V \setminus S$ para S) for menor ou igual ao número de arestas que conectam o subconjunto S aos demais nós de G.

Garantindo as condições estabelecidas para um grafo balanceado, o próximo passo é encontrar o *tour* Euleriano no grafo misto. Eiselt, Gendreau e Laporte (1995) apontam três fases para se determinar um *tour* em grafos mistos: (1) atribuir direções (orientações) a algumas arestas para que G se torne simétrico; (2) atribuir direções para as arestas que restaram; (3) definir o tour em G.

Ford e Fulkerson (1962) descrevem um algoritmo que transforma um grafo que não é simétrico em simétrico, conforme a etapa (1), descrita acima. Eiselt, Gendreau e Laporte, (1995) apresentam um algoritmo para orientar completamente um grafo simétrico, segundo a etapa (2). Na etapa (3) pode ser utilizado qualquer algoritmo para encontrar um *tour* de Euler em grafos direcionados, como já mencionado na seção 2.4.4. Assad e Golden (1995) apresentam um algoritmo para o PCCM em grafos simétricos para torná-los balanceados baseado em Evans e Minieka (1992). Edmonds e Johnson (1973) propuseram um algoritmo ótimo para balancear grafos mistos, orientando algumas arestas e acrescentando alguns arcos direcionadas ou não orientados ao grafo original.

Quando o grafo G não é Euleriano deve-se gerar um grafo aumentado G', que garanta as propriedades de unicursalidade, replicando um número suficiente de arestas e arcos de G para que o grafo resultante seja Euleriano e, por fim, resolver o PCCM determinando um aumento de custo mínimo (ASSAD; GOLDEN, 1995). Para tal caso, a literatura desenvolveu modelos de Programação Linear Inteira para tornar um grafo qualquer em Euleriano.

Um dentre estes modelos matemáticos formulados em programação linear inteira é a versão sugerida por Ahuja, Magnanti e Orlin (1993) para grafos mistos. Onde separa-se os componentes do grafo em dois grupos, um incluindo somente os nós com arcos direcionados, tal que $(i, j) \in A$ e outro contendo apenas os nós compondo as arestas não direcionadas em sua incidência, tal que $(i, j) \in E$.

$$Minimize \sum_{(i,j)\in A} c_{ij}\, x_{ij} + \sum_{(i,j)\in E} c_{ij}\, x_{ij} \tag{2.8}$$

$sujeito\ a$:

$$\sum_{j:(i,j)\in A} x_{ij} + \sum_{j:(i,j)\in E} x_{ij} = \sum_{j:(j,i)\in A} x_{ji} + \sum_{j:(j,i)\in E} x_{ji}\ ,\ \forall\, i \in N \tag{2.9}$$

$$x_{ij} + x_{ji} \geq 1,\quad \forall\,(i,j) \in E \tag{2.10}$$

$$x_{ij} \geq 1,\quad \forall\,(i,j) \in A \tag{2.11}$$

$$x_{ij}\ \text{é}\ inteiro,\quad \forall\,(i,j) \in A \tag{2.12}$$

$$x_{ij}\ \text{é}\ inteiro,\quad \forall\,(i,j) \in E \tag{2.13}$$

O modelo matemático é formulado fazendo x_{ij} ser uma variável que recebe o número de vezes que cada aresta/arco (i,j) será caminhada em *G* para um custo especifico c_{ij}. A função objetivo (2.8) minimiza o custo total do circuito. As restrições (2.9) garantem a conservação de fluxo, (2.10) forçam cada aresta a ser percorrida pelo menos uma vez durante o *tour* e (2.11) forçam cada arco a ser percorrido no mínimo uma vez durante o *tour*. As restrições (2.12) e (2.13) indicam que x_{ij} pertence ao conjunto dos números inteiros, tanto para arestas quanto para arcos.

Outra formulação matemática bastante conhecida em programação inteira para o PCCM com resolução exata é proposta por Christofides et al. (1983), que também apresentou em seu trabalho um algoritmo que garante a otimalidade do problema em redes mistas. O algoritmo é baseado em um algoritmo de *branch-and-bound* utilizando relaxação lagrangeana.

Encontrar um *tour* Euleriano em um grafo misto não Euleriano é uma abordagem difícil encontrada na literatura. Diferente dos casos não direcionados e direcionados, o PCCM é *NP*-completo, como provado por Papadimitriou (1976). Portanto, não se conhece resolução em tempo polinomial para o PCCM e, embora existam modelos bastante eficientes em problemas de pequeno e médio porte, na prática para aplicações em problemas de grande porte estes são ineficientes computacionalmente (PEARN; LIU, 1995).

A literatura relacionada ao PCCM dispõe de diversas abordagens heurísticas na busca por determinar soluções aproximadas para o caso. As principais heurísticas para o PCCM foram propostas por Edmonds e Johnson (1973) e por Frederickson (1979). Algumas modificações das heurísticas dos dois casos anteriores foram apresentadas por Pearn & Liu (1995). Pearn e Chou (1999) também divulgaram versões melhoradas das heurísticas de Edmonds e Johnson (1973) e Frederickson (1979); Corberán, Martí e Sanchis (2002) expuseram um algoritmo baseado em técnicas do procedimento GRASP (*Greedy Randomized Adaptive Search*

Procedure), Yaoyuenyong, Charnsethikul e Chankong (2002) propuseram uma heurística de busca usando um algoritmo de fluxo de custo mínimo e um algoritmo de caminho mais curto com custos ligeiramente modificados das arestas/arcos.

A abordagem clássica do PCC geralmente tem algumas premissas limitantes, por exemplo, por apenas considerar a rota projetada para um único veículo e por determinar as rotas em *loops* fechados, como foi apontado por Shafahi e Haghani (2015). Assim, o problema deixa de considerar algumas restrições que também são importantes de serem consideradas na modelagem matemática e que torna o problema mais prático e aplicável. Um desses casos se trata de considerar a capacidade dos veículos coletores que percorrem as ruas (arcos/arestas) e visto que estes podem não suportar a capacidade total de coleta do seu percurso, surge a necessidade de considerar mais de um veículo para realizar a coleta, este caso é uma variação do CPP, conhecido como Problema do Carteiro Chinês Capacitado.

2.5 PROBLEMA DO CARTEIRO CHINÊS CAPACITADO

O Problema do Carteiro Chinês Capacitado (PCCC) é um problema de otimização combinatória que é um caso especial do problema conhecido na literatura como Problema de Roteamento de Arcos Capacitados.

2.5.1 Problema de Roteamento de Arcos Capacitados

O Problema de Roteamento de Arcos Capacitados (PRAC) é um problema clássico de otimização combinatória, o qual foi introduzido por Golden e Wong (1981), que observaram que os problemas de roteamento em arcos eram pouco explorados com relação aos de roteamentos em nós, como o clássico Problema do Caixeiro Viajante (PCV). Os autores ainda exploraram diversas aplicações práticas e propuseram um modelo exato baseado em programação matemática e procedimentos aproximados para resolução do PRAC.

O PRAC é representado por um grafo $G = (N, E \cup A)$, sendo N o conjunto de nós, E o conjunto de arestas e A o conjunto de arcos. Em G existe um custo associado $c_{ij} > 0$ para todo $(i, j) \in E \cup A$ e uma demanda $d_{ij} \geq 0$ para todo $(i, j) \in E \cup A$. A demanda deverá ser coletada (ou entregue) por um conjunto de m veículos com capacidade restrita Q cada.

Três restrições são definidas para o roteamento de arcos capacitados: 1) cada veículo inicia e termina seu *tour* no nó n_0 chamado de depósito (ou garagem); 2) arestas e arcos com

demanda positiva e não-nula devem ser percorridos pelo menos uma vez por um único veículo; e 3) a demanda total coletada (ou entregue) por um *tour* não deve exceder a capacidade máxima do veículo (CORBERÁN; PRINS, 2010; FOULDS; LONGO; MARTINS, 2015; YAO et al., 2017). Além disso, para qualquer $(i, j) \in E \cup A$ com $d_{ij} = 0$, não é necessário que seja percorrido, o que significa, que não há demanda para coleta.

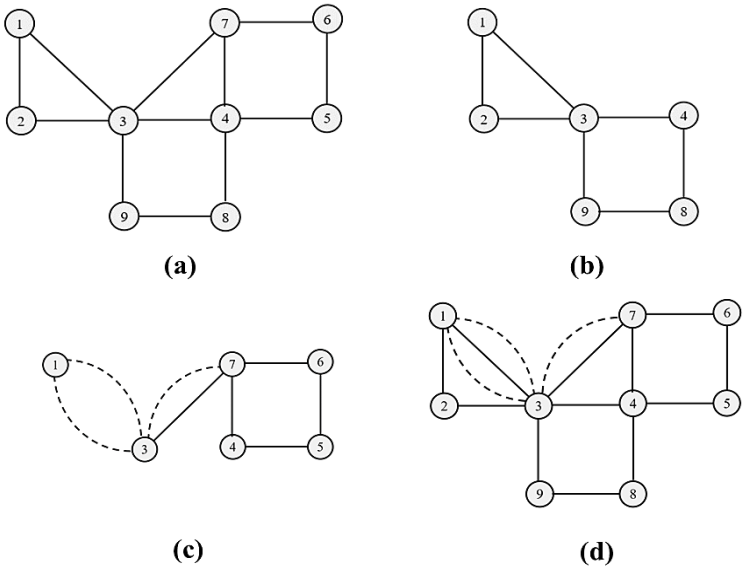
O objetivo do PRAC é, portanto, determinar um conjunto de *tours* com custo mínimo para os *m* veículos percorrendo todas as(os) arestas(arcos) com demanda positiva e não-nula no mínimo uma vez, respeitando a capacidade máxima Q de cada veículo.

No traçado dessa rota para cada veículo, algumas arestas/arcos serão percorridas assumindo a demanda e o custo da aresta, enquanto outras também serão percorridas, mas somente um custo de atravessá-la é contado. Isso ocorre quando uma aresta (uma rua, na prática) for percorrida mais de uma vez (YAO et al., 2017; CAMPBELL et al., 2018).

Uma aresta (i, j) é dita necessária quando ela for percorrida e atribuída uma demanda d_{ij} e um custo c_{ij} atrelado a essa aresta. Quando ela é percorrida e contado apenas o custo de cobri-la, dá-se o nome de *deadhead*. As arestas de *deadhead* são arestas fictícias que são atribuídas aos nós quando as arestas originais já foram percorridas, porém ainda é preciso atravessá-las novamente para concluir o *tour* (BELENGUER; BENAVENT, 2003; CAMPBELL et al., 2018).

Para ilustrar a tarefa de encontrar duas rotas em um grafo solucionado pelo PRAC, a Figura 11 apresenta um grafo não direcionado G (Figura 11.a) com um depósito no nó 1. Determina-se um *tour* para a rota 1 (Figura 11.b) e um *tour* para a rota 2 (Figura 11.c), onde as arestas necessárias a serem percorridas e atendidas são desenhadas em linhas contínuas e as arestas de *deadhead* são desenhadas em linhas tracejadas. No fim, apresenta-se uma solução viável para o PRAC com duas rotas em G (Figura 11.d).

Figura 11 – Solução em um grafo para o PRAC



Fonte: O autor (2020).

Dror (2012) afirmou que existem duas versões para o PRAC, ambas relacionadas à quantidade de veículos na modelagem. Na primeira versão, *m* é fixo e o modelo irá traçar as rotas dado o número de veículos. Já na segunda versão, o tamanho da frota é uma variável de decisão, cuja solução da modelagem determinará o número de veículos necessários a percorrer todos as arestas no menor custo e sem exceder as capacidades dos veículos. Sendo a determinação da solução ótima para o primeiro caso um problema *NP-difícil*, então, aumentar o número de variáveis do problema para o segundo caso, torna o trabalho ainda mais difícil, o que justifica o crescimento do número de heurísticas para sua resolução (VECCHI et al., 2016).

Uma outra variação do PRAC é quando todas as demandas presentes nas(os) arestas(arcos) precisam ser atendidas(os), isto é, toda(o) aresta(arco) têm demanda positiva e não-nula ($d_{ij} > 0$ para todo $(i, j) \in E \cup A$). Logo, o problema se reduz à versão capacitada do PCC, denominada de Problema do Carteiro Chinês Capacitado (PCCC) (ASSAD; GOLDEN, 1995).

2.5.2 Problema do Carteiro Chinês Capacitado

O PCCC foi introduzido por Christofides (1973). O PCCC é um caso especial do PRAC quando se têm todas as arestas/arcos apresentando demanda positiva e não-nula. Neste caso, todas as arestas/arcos devem ser percorridas pelo menos uma vez por um conjunto de *m* veículos com uma capacidade conhecida *Q* no menor custo possível para atender toda a demanda.

Quando a capacidade de um veículo é maior ou igual a soma de todas as demandas ($Q \geq \sum_i \sum_j d_{ij}$) o PCCC é reduzido ao PCC (GOLDEN; WONG, 1981).

O PCCC tem complexidade *NP-difícil* como foi provado por Golden e Wong (1981). Dror (2012) ainda expõe que o PRAC/PCCC é *NP-difícil* para ambos os casos não direcionado e direcionado, por derivar do clássico Problema de Roteamento de Veículos (PRV). Então, qualquer que seja o tipo de grafo, a resolução para o PCCC terá nível de complexidade computacional difícil de ser resolvido.

Entretanto, existem alguns modelos que resolvem o PCCC de forma ótima. Um deles é o modelo de Programação Linear Inteira Mista (PLIM) proposto por Golden e Wong (1981). O modelo é apresentado a seguir.

Dado um grafo $G = (N, E)$, onde N é o conjunto de nós e E o conjunto de arestas, o objetivo é encontrar um conjunto de *tours* para um conjunto de veículos $M = \{1, 2, ..., m\}$, em que cada veículo possui capacidade Q e deve atender a demanda existente em G. Este problema é formulado como um problema de fluxo de tal forma:

$$Minimize \quad \sum_{i \in N} \sum_{j \in N} \sum_{p \in M} c_{ij} x_{ij}^{p} \tag{2.14}$$

$sujeito\ a$:

$$\sum_{j \in N} x_{ji}^{p} - \sum_{j \in N} x_{ij}^{p} = 0\,, \qquad \forall\, i \in N, p \in M \tag{2.15}$$

$$\sum_{p=1}^{m} \left(l_{ij}^{p} + l_{ji}^{p}\right) = \left\lceil \frac{d_{ij}}{Q} \right\rceil\,, \qquad \forall\, (i,j) \in E \tag{2.16}$$

$$x_{ij}^{p} \geq l_{ij}^{p}\,, \quad \forall\, (i,j) \in E, p \in M \tag{2.17}$$

$$\sum_{i \in N} \sum_{j \in N} l_{ij}^{p} d_{ij}^{p} \leq Q\,, \quad \forall\, p \in M \tag{2.18}$$

$$\sum_{j \in N} f_{ij}^{p} - \sum_{j \in N} f_{ji}^{p} = \sum_{j \in N} l_{ij}^{p}\,, \quad \forall\, i \in N/\{1\}, p \in M \tag{2.19}$$

$$f_{ij}^{p} \leq (n^2)\, x_{ij}^{p}\,, \qquad \forall\, (i,j) \in E,\ p \in M \tag{2.20}$$

$$f_{ij}^{p} \geq 0\,, \quad p \in M \tag{2.21}$$

$$l_{ij}^{p}, x_{ij}^{p} \in \{0,1\}\,, \forall\, (i,j) \in E, p \in M \tag{2.22}$$

Para o modelo, tem-se os seguintes dados: c_{ij} representa o comprimento do arco (i, j); d_{ij} é a demanda a ser atendida no arco (i, j); Q representa a capacidade de cada veículo; l_{ij}^{p} é

uma variável binária que assume 1 quando o veículo *p* atender a demanda presente em (i, j), e assume 0, caso contrário; f_{ij}^{p} é uma variável de fluxo que apresenta valor positivo quando x_{ij}^{p} = 1.

A função objetivo (2.14) minimiza o comprimento total percorrido pelos veículos. A restrição (2.15) assegura a conservação do fluxo para cada veículo. A restrição (2.16) garante que o atendimento dos veículos será considerado em apenas uma de suas passagens pelo arco (i, j). A restrição (2.17) garante que a aresta (i, j) será percorrida pelo veículo *p* somente se ele atravessar o arco (i, j). A restrição (2.18) permitirá que a capacidade máxima de cada veículo não seja ultrapassada. As restrições (2.19), (2.20) e (2.21) formam um conjunto de restrições que foram empregadas para assegurar a eliminação de sub rotas, ou seja, rotas desconexas. E por fim, (2.22) definem as classes das variáveis e parâmetros de entrada.

O modelo de Golden e Wong (1981) fomenta uma série de aplicações práticas e será abordado mais à frente como um modelo matemático, que resolve o PCCC de forma exata, garantindo que a solução seja ótima. Mesmo sendo um problema difícil de solucionar, mas vale o esforço computacional para comparar com as abordagens que não garantem a otimalidade.

O PRAC/PCCC engloba a maioria das aplicações reais das vias urbanas com casos relacionados à coleta ou entrega de produtos (VECCHI et al., 2015). Assim, é válido buscar na literatura algumas das principais contribuições dessa metodologia nos contextos práticos e em diversas abordagens para sua resolução.

3 REVISÃO DA LITERATURA

O presente capítulo apresenta os principais e relevantes trabalhos encontrados na literatura que desenvolveram abordagens heurísticas para resolução do PRAC e suas variações, seja em contextos práticos, seja em testes computacionais. E em seguida, são expostas algumas aplicações do PRAC/PCCC na coleta de lixo urbano encontradas na literatura.

3.1 APLICAÇÕES PRÁTICAS

As aplicações do PRAC em contextos do dia a dia são bem abrangentes e aqui apresenta-se algumas das principais.

Gáspár e Bencze (2016) trabalharam na aplicação (distribuição) de sal em redes rodoviárias. Espalhamento ou disseminação de sal é uma atividade cotidiana no inverno em países com clima frio e sua aplicação evita que as estradas fiquem escorregadias no período da geada, do gelo ou da neve. Os autores focaram em otimizar a eficiência e o tempo das políticas de armazenamento de sal e de degelo nas estradas em função dos aspectos ambientais, econômicos e de segurança no trânsito. A metodologia proposta foi modelada como um PRAC com diversas variáveis abordando o ponto de vista ambiental e econômico.

Usberti, França e França (2008) assimilaram o PRAC com o Problema de Roteamento de Leituristas (PRL). Esse problema é do interesse das companhias distribuidoras de energia elétrica, água e gás que realizam medição periódica do consumo de seus clientes. Os autores apresentaram duas reduções desses problemas combinatórios: a primeira mapeia qualquer instância do PRL em uma instância do PRAC (tempo polinomial), possibilitando resolvê-lo por meio de algoritmos já existentes para o PRAC. Já a segunda redução mapeia qualquer instância do PRAC em uma instância para o PRL (em tempo polinomial). Isto prova que o PRL também é *NP-difícil*, o que fundamenta o desenvolvimento de heurísticas para sua solução.

Chen et al. (2014) estudaram alguns métodos de otimização para um problema de roteamento encontrado em operações de manutenção diária de uma rede rodoviária. Entre eles, os autores formularam o problema como uma variação do PRAC, considerando serviços estocásticos e tempos de viagem em segmentos das estradas. A resolução partiu de um algoritmo Busca Adaptativa em Vizinhança de Grande Porte.

O gerenciamento de detritos (ou entulhos) é uma preocupação após qualquer grande desastre, no qual, o processo de remover os detritos leva meses ou até anos para terminar, na maioria dos casos. Pramudita e Taniguchi (2014) trataram da coleta de detritos após desastres

como um caso do PRAC. Um modelo em programação matemático foi apresentado baseado nas restrições do veículo, porém adicionado uma restrição de possibilidade de acesso, devido algumas ruas ficarem obstruídas. Foi proposto uma meta-heurística de Busca Tabu para resolução do PRAC para o problema da operação de coleta de detritos. O modelo foi aplicado em um caso prático baseado na Região Metropolitana de Tóquio.

Ting e Tsai (2018) apresentam um algoritmo de Otimização de Colônia de Formigas com uma abordagem chamada *Path Relinking* (ACOPR) para solucionar o PRAC. A ACOPR proposta foi testada em um conjunto de 181 instâncias da literatura para avaliar sua eficácia e comparar com as meta-heurísticas de melhor desempenho existentes. Os resultados computacionais mostram que o ACOPR é competitivo com os algoritmos meta-heurísticos do estado da arte. Os autores viram uma aplicabilidade do problema em serviços de limpeza de ruas e utilizaram esta aplicação para mostrar a eficiência do ACOPR proposto.

Li et al. (2018) aplicaram um caso prático de monitoramento de tráfegos com o uso de veículos aéreos não tripulados (*drones*). Neste artigo, o PRAC é combinado com o Problema de Roteamento de Estoque (PRE) para criar o problema de programação de veículos aéreos não tripulados com demandas incertas, onde um modelo de programação inteira mista foi empregado, baseado no modelo do PRAC proposto por Golden e Wong (1981) e a partir deste acrescentado diversas características típicas do problema. Um método de solução baseado em *local branching* é desenvolvido para resolver o modelo. O problema foi aplicado em um estudo de caso ao modelo de tráfego rodoviário em Xangai, China.

Kloimüllner et al. (2015) utilizaram a heurística construtiva *Cluster-First Route-Second* para resolução do problema de equilibrar sistemas de compartilhamento de bicicletas na cidade de Viena, Áustria. A abordagem utilizada atribuiu estações aos veículos e, em seguida, resolveu o problema de acordo com o roteamento por um algoritmo exato que utiliza a decomposição de Benders baseada em lógica (*logic-based Benders decomposition*). Os resultados resolveram instâncias com até 60 estações, o que não era possível nas definições de problemas anteriores.

Schuijbroek, Hampshire e Van Hoeve (2017) também trataram do problema de bicicletas compartilhadas, onde o principal desafio deste sistema é o reequilíbrio das bicicletas ao longo do tempo. Para isso, os autores mostraram dois aspectos conhecidos na literatura: determinar os requisitos de nível de serviço em cada estação de compartilhamento de bicicletas e projetar rotas viáveis para os veículos reequilibrar a estação (de bicicletas). Assim, foi proposto uma nova heurística *Cluster-First Route-Second*, em que um problema de *cluster* considera simultaneamente a viabilidade do nível de serviço e os custos aproximados de roteamento. Foram apresentados resultados computacionais para dados reais de duas cidades

dos Estados Unidos, que mostraram que a heurística é mais aplicável, no contexto prático, do que um modelo de Programação Inteira Mista e uma abordagem de Programação Restrita encontradas na literatura.

O trabalho de Ghorpade e Rangaraj (2019) considerou uma estratégia de planejamento de ciclo alternativo para movimentos no transporte ferroviário de carga na Índia. O sistema foi modelado como o Problema de Coleta e Entrega (*Pickup and Delivery Problem*). Foi apresentado uma formulação matemática exata e uma heurística *Order-First Split-Second* modificada. O (*giant tour*) foi construído atribuindo um valor a cada nó com base no caminho guloso desse local. O *Splitting Algorithm* formou vários ciclos, satisfazendo a restrição máxima de tempo de ciclo do *tour* gigante (*giant tour*). O algoritmo proposto foi testado nos dados de movimentos dos trilhos de carga na Índia, onde os autores conseguiram quantificar várias melhorias ao sistema de trem indiano.

Motivados pelos serviços de aspersão de água em uma grande cidade na China, Yu et al. (2019) estudaram o PRAC com Entrega Fracionada (PRACEF), uma extensão natural do PRAC que permite servir uma aresta usando vários veículos, ou seja, a demanda pode ser dividida. A opção de dividir é útil quando a demanda for maior que a capacidade do veículo, isto pode reduzir potencialmente o custo total da viagem. Os autores trataram o PRACEF para grafos mistos. Uma formulação matemática e um algoritmo *Forest-Based Tabu Search* foi proposto para resolver o problema. Experimentos numéricos mostraram que o algoritmo produz soluções de alta qualidade quando comparado com as abordagens da literatura.

3.2 METODOLOGIAS PARA RESOLUÇÃO DO PRAC

Devido ao PRAC e suas variações serem problemas *NP-difíceis*, existem diversos algoritmos aproximados, heurísticos e meta-heurísticos na busca por sua resolução. Esta seção aborda alguns trabalhos encontrados na literatura nos últimos 20 anos.

Lenis e Rivera (2018) indicam uma nova variante para o PRAC, no qual a função objetivo torna-se um objetivo cumulativo calculado como a distância percorrida multiplicada pela carga do veículo. Para isso, uma abordagem meta-heurística foi proposta com base na hibridização de três procedimentos: GRASP (*Greedy Randomized Adaptive Search Procedure*), VND (*Variable Neighborhood Descent*) e um modelo de cobertura de conjuntos. Dois algoritmos construtivos foram incorporados. O primeiro algoritmo construtivo baseia-se na heurística *Route-First Cluster-Second*, que constrói uma rota que atravessa todas as arestas necessárias sem considerar as restrições de capacidade. Em seguida, o algoritmo verifica a rota

levando em consideração as demandas e a capacidade total para ele retornar ao depósito. Um segundo algoritmo construtivo foi considerado para evitar que veículos em seu percurso percorram rotas muito longas. A proposta foi testada em algumas instâncias de *benchmark* para o PRAC.

Porumbel et al. (2017) apresentaram um método que combina Geração de Colunas (CG) e Busca Local Iterada (*Iterated Local Search* – ILS) como alternativa para a resolução do PRAC. Um dos objetivos foi integrar ao ILS informações de dualidade que sustentam o paradigma do CG.

Longo, De Aragão e Uchoa (2006) transformaram o PRAC em um Problema de Roteamento de Veículos Capacitados (PRVC), os autores utilizaram uma abordagem conhecida pela literatura, porém por esta ser considerada inviável, então foi proposto uma nova transformação que reduz a instâncias do PRVC para haver soluções viáveis para o PRAC. Os resultados mostraram ser significativamente melhores do que os métodos exatos criados para resolução do PRAC.

Mais tarde, Foulds, Longo e Martins (2015) descreveram um método compacto para transformar instâncias do PRAC em instâncias do PRVC por meio de um processo adaptado de um algoritmo *branch-cut-and-price*. Os experimentos computacionais produziram limites inferiores úteis em tempo computacional razoável para muitas instâncias numéricas desafiadoras da literatura. Além disso, algumas dessas instâncias previamente abertas foram resolvidas na otimalidade pela primeira vez.

Brandão e Eglese (2008) utilizaram uma abordagem heurística baseado em Busca Tabu, no qual os resultados computacionais demonstraram a alta qualidade do algoritmo para fornecer soluções ao PRAC em um tempo computacional razoável.

Chen, Hao e Glover (2016) desenvolveram uma nova abordagem meta-heurística híbrida para resolver efetivamente o PRAC, empregando um procedimento chamado de *Randomized Tabu Thresholding Procedure* (RTTP) integrado a um procedimento para explorar regiões viáveis e inviáveis. O trabalho apresentou diversos estudos experimentais, que mostraram o quanto a meta-heurística é altamente viável e capaz de identificar rapidamente os melhores resultados para quase todas as instâncias de *benchmark* disponíveis na literatura para o PRAC.

Santos, Coutinho-Rodrigues e Current (2010) apresentaram uma meta-heurística baseada em Otimização de Colônia de Formigas (ACO) para resolução do PRAC. As principais contribuições no ACO foram modificações na população inicial, na regra de decisão *ant* e no procedimento de Busca Local.

Gouveia, Mourão e Pinto (2010) estudaram o Problema de Roteamento de Arcos Capacitados Mistos (PRACM). O PRACM considera um grafo com arestas e arcos. Foi descrito um modelo compacto baseado em fluxo para o PRACM, que devido à enorme quantidade de variáveis e restrições associadas, dois novos modelos foram apresentados, um modelo de Programação Linear Inteira Mista (PLIM) e um modelo agregado incorporado ao primeiro. O valor das variáveis relaxadas do modelo PLIM e do modelo agregado foram iguais para diversos conjuntos de instâncias, conforme os autores.

O PRACM também foi estudado por Masran e Ramli (2019) considerando restrições de capacidade múltipla. Os autores apresentaram uma heurística construtiva baseada em métodos de inserção para o problema. Em seguida, abordaram algumas semelhanças e diferenças entre a heurística proposta e o modelo matemático de Gouveia, Mourão e Pinto (2010). Experimentos computacionais foram realizados usando grafos com 10, 25 e 50 nós para grafos não direcionados, dirigidos e mistos aleatoriamente, que mostraram bons resultados.

Feng et al. (2010) apresentaram um Algoritmo Memético Probabilístico (PMA) para o PRAC equipado a um mecanismo adaptado para controlar o grau de exploração global diante da exploração local enquanto a busca progride. O algoritmo memético melhorado para o PRAC demonstrou desempenhos plausíveis com esquemas para a Busca Local. O PMA, entre outros métodos de busca meta-heurística, demonstrou alcançar desempenhos competitivos na resolução de PRAC, variando de pequeno a médio porte.

Willemse e Joubert (2016) apresentaram o Problema de Roteamento de Arco com Capacidade Mista sob Restrições de Tempo com Instalações Intermediárias (PRACMRTII), uma extensão do PRAC com aplicação na coleta de lixo municipal. Os autores avaliaram quatro heurísticas construtivas que apresentassem soluções viáveis para o PRACMRTII com o objetivo de minimizar o custo total ou minimizar o tamanho da frota. As heurísticas foram adaptadas de *Path-Scanning* e *Improved-Merge* para o PRACM e comparadas com duas heurísticas *Route-First Cluster-Second* para o PRACMRTII. No final do estudo, a heurística *Path-Scanning* se mostrou mais eficiente e robusta.

Babaee Tirkolaee et al. (2016) desenvolveram um novo modelo matemático para o PRAC robusto para a coleta de lixo. O PRAC robusto visa minimizar a distância percorrida de acordo com a incerteza de demanda nas arestas, assim, o modelo robusto apresenta uma abordagem de otimização robusta por intervalo. Para resolver o problema os autores apresentaram um algoritmo meta-heurístico híbrido com base em no algoritmo *Simulated Annealing* e em um algoritmo heurístico. Os resultados obtidos com o algoritmo proposto foram

comparados com os resultados do método exato, onde os resultados mostraram que o desempenho da meta-heurística híbrida proposta foi aceitável.

Martinelli, Poggi e Subramanian (2013) trataram das abordagens exatas e heurísticas para o PRAC com ênfase em melhorar os limites inferiores e superiores de instâncias de grande porte (grafos com mais de 200 vértices e 300 arestas). Para tanto, no limite inferior, sugeriram explorar a velocidade de uma heurística chamada *dual ascent* para gerar cortes de capacidade; e no limite superior, aplicaram um procedimento de Busca Local Iterada para instâncias do PRVC, que é transformado a partir das instâncias originais do PRAC.

Arakaki e Usberti (2019) abordaram uma nova heurística, denominada *path-scanning*, para o PRAC, a partir da introdução do conceito de regra de eficiência (*PS-Efficiency*). Dada a localização atual do veículo, sua distância percorrida e a quantidade de demanda atendida, a regra de eficiência seleciona as arestas mais promissoras a serem atendidas em seguida. A regra de eficiência baseou-se em como cada aresta afetaria a eficiência atual da rota calculada como a proporção da demanda atendida pela distância percorrida. A partir de um conjunto de instâncias de *benchmark*, os experimentos computacionais revelaram que a heurística proposta foi superior a todas as heurísticas *path-scanning* encontradas na literatura.

Van Bevern, Komusiewicz e Sorge (2017) aplicaram a heurística construtiva *Route-First Cluster-Second* para resolução do PRAC e do PRAC Misto e com Vento (PRACMV). Os autores primeiro calcularam um *tour* gigante (*giant tour*) aproximado para o PRACMV e posteriormente o dividiram em *sub tours*, cada um servido por um veículo com capacidade *Q* e um depósito, por meio de uma abordagem aproximada considerando pesos nos componentes fracamente conectados do subgrafo pelos arcos/arestas com demandas positivas. A abordagem proposta foi comparada com outras duas heurísticas da literatura, mostrando ser superior.

Amberg, Domschke e Voβ (2000) apresentaram o PRAC com vários centros (M-PRAC), onde o objetivo é encontrar rotas iniciando de múltiplos depósitos. Os autores utilizaram a abordagem *Route-First Cluster-Second* para o M-PRAC em grafos não direcionados, onde na primeira fase o problema foi modelado como uma versão modificada do problema *Capacitated Minimum Spanning Tree* (CMST) e uma heurística foi utilizada para gerar soluções iniciais. Na segunda fase, aplicaram as meta-heurísticas *Simulated Annealing* e Busca Tabu para melhorar as rotas geradas na fase anterior.

Prins, Lacomme e Prodhon (2014) realizaram a primeira revisão da literatura para o PRAC e PRVC utilizando métodos *Order-First Split-Second*. Esta abordagem é conhecida por determinar primeiro grupos de clientes em comum com a capacidade dos veículos e em seguida traçam uma rota para cada grupo determinado. Ela se mostra mais relevante do que a *Route-*

First Cluster-Second quando se tem um conjunto de clientes em comum que precisa ser atendido por um veículo específico, devendo se determinar posteriormente apenas a sequência de clientes atendidos. Na revisão foram analisados os métodos utilizados para o *Order-First Split-Second*, que reuniram mais de 70 referencias para resolução dos problemas tratados.

Wang et al. (2015) abordaram o Problema de Roteamento de Arcos Capacitados Incertos (PRACI), que considera as demandas, os custos e as arestas como incertas. Para resolução do problema, os autores apresentaram uma nova abordagem de otimização, chamada *Estimation of Distribution Algorithm* (EDA) com Busca Local Estocástica (BLE). O método proposto integrou a EDA ao BLE em duas fases para minimizar o custo total máximo em um conjunto de cenários diferentes. Durante a fase de melhoria do cromossomo aplicou-se o algoritmo *Split Procedure* para gerar soluções ótimas. Resultados experimentais para um total de 55 instâncias de problemas mostraram que a abordagem superou os algoritmos do estado da arte abordado.

Arakaki e Usberti (2018) avaliaram o Problema de Roteamento de Arcos Capacitados Abertos (PRACA). O PRACA difere do PRAC, uma vez que, o PRACA não considera um depósito e as rotas não estão restritas a formar circuitos fechados. Um Algoritmo Genético Híbrido (HGA) com procedimentos de viabilização e Busca Local foi proposto para o PRACA. Experimentos computacionais para um conjunto de instâncias de *benchmark* revelaram que o HGA proposto alcançou os melhores limites superiores para quase todas as instâncias. Os resultados foram comparados com outros métodos heurísticos da literatura.

Vidal (2017) explorou uma decomposição estrutural de vizinhança para os Problemas de Roteamento em Arcos, na qual as decisões sobre orientações de deslocamento são tomadas de maneira ótima como parte dos procedimentos de avaliação de vizinhos, por meio de estruturas de memória, programação dinâmica bidirecional e limites inferiores. Para investigar as capacidades de melhoria das vizinhanças, foi integrado duas meta-heurísticas clássicas: a Busca Local Iterada e a Busca Genética Híbrida Unificada. Foram realizados experimentos computacionais em diversas variantes dos Problemas de Roteamento em Arcos, incluindo o PRAC e o Problema de Roteamento Geral Capacitado Misto (PRGCM), que tiveram um estudo aprofundado para o caso com penalidades de *tour* (*Turn Penalties*), que são considerados problemas críticos. Os experimentos computacionais levaram a soluções de alta qualidade em 18 conjuntos de *benchmark* para esses problemas, com um total de 1.528 instâncias.

Tirkolaee, Mahdavi e Esfahani (2018) apresentam um modelo matemático para o PRAC com restrição de periodicidade, considerando múltiplas viagens e tempos de trabalho de motoristas e tripulantes. O modelo proposto objetivou minimizar a distância total percorrida e o custo total de uso de veículos durante um período de planejamento. Na resolução foi

desenvolvido um algoritmo *Simulated Annealing* híbrido com base em uma heurística construtiva e uma eficiente equação de resfriamento. Os autores provaram que o desempenho do algoritmo proposto é admissível quando comparado com o método da solução exata.

3.3 APLICAÇÕES EM COLETA DE LIXO

Dado a problemática da coleta de lixo, que atingiu uma preocupação a nível global, os trabalhos desta revisão fomentam as novas abordagens em Problemas de Roteamento em Arcos para sua resolução.

Ghiani et al. (2005) realizaram um estudo sobre a coleta de resíduos sólidos na cidade de Castrovillari, Itália. Os autores verificaram que o problema da coleta de lixo no município se tratava de um Problemas de Roteamento em Arcos (PRA). Assim, criaram uma heurística construtiva chamada *cluster-first route-second*. A heurística foi implementada em um sistema informatizado, que garantiu diversos benefícios como: redução na distância total percorrida, redução no tempo de trabalho e eliminação das horas extras presentes na atividade, onde tais resultados permitiram alcançar uma redução de cerca de 8% no custo total do processo.

Mourão e Amado (2005) apresentaram um método heurístico para gerar soluções viáveis para o PRAC estendido em grafos mistos, inspirado no problema de coleta de lixo doméstico em Lisboa, Portugal. A heurística foi comparada, computacionalmente, com algumas heurísticas existentes bem conhecidas, generalizadas para um PRAC estendido. Foram abordados problemas com tamanhos significantes de nós, que apresentaram valores de *gap* variando entre 0,8% e 3% e pequenos tempos de execução entre 0 segundos e 12 minutos.

Bautista, Fernández e Pereira (2008) aplicaram uma metodologia para a solução de um problema de coleta de lixo urbano no município de Sant Boi de Llobregat, Espanha. Os autores utilizaram a heurística Colônias de Formigas e algumas heurísticas de vizinhança combinadas (entre eles, métodos construtivos, baseados no vizinho mais próximo e na inserção mais próxima, com uma Busca Local que explorava vários bairros). Os autores obtiveram uma redução do comprimento total das rotas em torno de 35% e 37% para cada heurística utilizada, respectivamente, comparada com coleta de lixo urbano do município estudado.

Buscando apresentar um roteiro da literatura na área dos Problemas de Roteamento de Veículos para Coleta de Resíduos (PRVCR), Han e Ponce Cueto (2015) expuseram uma revisão bibliográfica, considerando os principais modelos e métodos de roteamento em arcos e nós para resolução do PRVCR. Os autores concluíram, que as pesquisas atuais sobre o PRVCR se concentram mais na melhoria da eficiência das práticas de coleta de lixo, minimizando a

distância, os custos logísticos e o tempo. O trabalho mostrou que existem pesquisas voltadas na melhoria das rotas de veículos de coleta de lixo, e outras preocupadas com a determinação da localização mais adequada das instalações de descarte e a localização dos depósitos de coleta.

Rodrigues e Soeiro Ferreira (2015) descreveram três casos em que é possível aplicar os problemas de roteamento de coleta de lixo na prática, com foco na cidade de Monção, Portugal, envolvendo vários locais de descarte, alguns com um número limitado de descartes por dia e usando veículos diferentes. Uma modelagem matemática foi proposta para cada caso e resultados computacionais foram testados. Os autores ainda forneceram uma contribuição para a literatura do PRAC, devido a seus recursos adicionais relacionados à frota heterogênea e a aterros múltiplos limitados. Os resultados computacionais foram aplicados com base em instâncias adaptadas da literatura e para o caso de Monção. Concluiu-se que soluções ótimas são alcançadas sempre que o número de viagens não exceder 5.

Vecchi et al. (2016) descreveram uma abordagem sequencial em três fases para resolução do problema de otimização de rotas de caminhões para a coleta de resíduos sólidos. Na primeira fase ocorre o agrupamento de arcos baseado em um modelo adaptado do problema das *p*-medianas; na segunda fase se desenvolveu um modelo PLIM para a solução do PRAC; e na terceira fase utilizou-se um algoritmo de Hierholzer adaptado para o sequenciamento dos arcos obtidos na fase anterior. A metodologia empregada foi aplicada para dois casos reais da cidade de Campo Mourão, Brasil. O primeiro caso proporcionou uma redução de 1,5% na distância total percorrida (economia de US $ 3.825/ano) e o segundo permitiu uma redução da distância total percorrida em 7,5% (economia de US $ 4.146/ano).

O trabalho de Wøhlk e Laporte (2018) foi motivado por um problema de coleta de lixo na Dinamarca para 6 munícipios, o qual desenvolveram uma heurística rápida chamada FastCARP para a solução do PRAC em larga escala, com ou sem restrições de duração. O procedimento inicia com um pré-processamento, onde o FastCARP cria um tour gigante (*giant tour*), divide o grafo em distritos por meio de um algoritmo *Split Procedure* e constrói rotas dentro de cada distrito. Em seguida, mescla (*Merge*) iterativamente e divide (*Split*) os distritos adjacentes e otimiza as rotas. A heurística foi testada para 264 instâncias de *benchmark* contendo até 11.640 nós, 12.675 arestas, 8581 arestas necessárias e 323 veículos. O FastCARP foi comparado com uma heurística alternativa chamada BASE e com vários algoritmos *Path-Scanning*.

Motivados por um problema real de coleta de lixo na Dinamarca, Wøhlk e Laporte (2019) resolveram um problema de vários períodos, envolvendo vários tipos de lixo denominado de *frações* (resíduos orgânicos, papel, papelão, vidro, etc.). Os autores

desenvolveram uma heurística multi-fases: (1) cria-se pequenos distritos de coleta; (2) os distritos são designados por dias úteis específicos; (3) os distritos são equilibrados; e (4) rotas de coleta são criadas para cada distrito e cada fração de resíduos por meio da heurística F_{AST}CARP de Wøhlk e Laporte (2018). O problema foi considerado para escalas muito grandes (instâncias entre 26 e 11.656 nós, 33 e 12.691 arestas, 19 e 8.651 arestas necessárias e 2 e 54 veículos). Os testes computacionais indicaram que a coordenação dos dias da coleta gerou um aumento de 12,4% dos custos de roteamento e um aumento de 9,1% no número de veículos.

Silva, Lins e Xavier (2019b) apresentaram um programa simples para resolução do PCCD atuando na determinação de rotas mínimas para veículos coletores de lixo na cidade de Recife, Brasil. O programa desenvolvido pelos autores recebe uma matriz de distâncias do grafo pelo *software OpenStreetMap* e determina a rota ótima em 3 etapas: escreve o modelo matemático que resolve o PCCD, resolve por um *solver* eficiente e traça a rota em um grafo Euleriano a partir do algoritmo de *Fleury* modificado para grafos direcionados. O programa foi implementado em linguagem de programação *Python*, aplicado em dois bairros da cidade e comparado com as rotas atuais da empresa de coleta de lixo municipal. Os resultados mostraram uma redução da distância total de 12,7% para o primeiro bairro e 8,86% para o segundo bairro.

4 METODOLOGIA

No presente capítulo é apresentado os procedimentos metodológicos desenvolvidos no trabalho, a classificação da pesquisa, descrição detalhada do método empregado com a definição dos *softwares* e fórmulas utilizados e detalhamento do procedimento heurístico com foco nos quatro passos estabelecidos para se determinar as rotas viáveis para veículos coletores.

4.1 CLASSIFICAÇÃO DA PESQUISA

Pela natureza do objetivo deste trabalho, a pesquisa é aplicada. Para Gerhardt e Silveira (2009) a pesquisa aplicada tem o objetivo de gerar conhecimentos com finalidade de aplicação prática, voltados a explicação de problemas específicos, envolvendo verdades e interesses locais. A forma de abordagem deste trabalho é quantitativa, uma vez que são usados dados mensuráveis e numéricos para resolução da pesquisa. Quanto ao objetivo, a pesquisa caracteriza-se como um estudo exploratório. Onde busca-se maior familiaridade com o problema envolvido, de modo a explicitá-lo (GIL, 2010).

O trabalho é desenvolvido através do método de modelagem. A modelagem utiliza-se de técnicas matemáticas para descrever o funcionamento de um sistema produtivo ou parte deste. A modelagem pode se definir como um modelo de uma representação de um sistema real, tornando-se essencial para o processo de tomada de decisão, visto que reúne dados como restrições e variáveis para processar as informações de um problema fornecendo uma solução otimizada para o cenário em estudo (GANGA, 2012).

4.2 DELINEAMENTO DO MÉTODO EMPREGADO

Inicialmente, foi realizado um levantamento bibliográfico sobre os principais temas utilizados neste trabalho: grafos Eulerianos, Problema de Roteamento em Arcos, Problema do Carteiro Chinês, Problema de Roteamento de Arcos Capacitados e Problema do Carteiro Chinês Capacitado (PCCC) com foco na coleta de lixo urbano. Este levantamento foi importante para o embasamento teórico e fundamentos relevantes à pesquisa.

O principal método empregado neste trabalho é o desenvolvimento de um algoritmo heurístico para resolução do PCCC em etapas bem definidas que serão descritas na seção 4.3. A fim de verificar a sua eficiência, os resultados da heurística proposta foram comparados com os resultados do modelo matemático proposto por Golden e Wong (1981) descrito na subseção

2.5.2. O objetivo é comparar as soluções da heurística e do modelo exato com relação ao tempo computacional, o tamanho do grafo quando se aumenta o número de nós e arcos e variando a capacidade máxima dos veículos.

O procedimento proposto se trata de uma heurística construtiva do tipo *Route-First Cluster-Second*. Segundo Van Bevern, Komusiewicz e Sorge (2017) o método *Route-First Cluster-Second* aborda primeiro a construção de um *tour* gigante, que cobre todos os arcos com demanda positiva do grafo balanceado e, em seguida, dividi-lo em grupos (*clusters*) satisfazendo as restrições de capacidade dos veículos.

Prins, Lacomme e Prodhon (2014) afirmam que a abordagem *Route-First Cluster-Second* é a mais comum para resolução do PRAC e suas variações (no caso, como o PCCC). As principais razões para esta afirmação são: por existir um espaço de solução menor para Problemas de Roteamento em Arcos do que os problemas em nós; flexibilidade, dado que muitas restrições podem ser tratadas; e eficiência, onde as meta-heurísticas mais comuns podem chegar a otimalidade com mais facilidade.

Para modelagem e validação da heurística foi utilizado dados reais de bairros para a coleta de lixo urbano na cidade de Recife, Pernambuco, Brasil. As informações sobre as rotas de coleta de lixo urbano e a capacidade máxima dos veículos foram fornecidas pela Secretaria de Infraestrutura e Serviços Urbanos de Recife por meio da empresa Autarquia de Manutenção e Limpeza Urbana do Recife (EMLURB), atual responsável pela coleta de lixo urbano na cidade. Foram escolhidos dois bairros onde o lixo é coletado para serem representados através de grafos. Os dados geográficos foram coletados a partir do uso do *software OpenStreetMap*.

O *OpenStreetMap* é um *software* cooperativo *online*, licenciado pela *Open Database License* (ODbL) (OPEN DATA COMMONS, 2019) de código aberto que apresenta uma visualização em formato de mapa de todo o globo terrestre com diversos dados vetoriais (OPENSTREETMAP, 2019).

No *OpenStreetMap* é possível selecionar bairros e por meio de uma modelagem especial utilizando atributos, transforma-se pontos de ruas em nós, segmento de ruas (caminhos) em arestas e locais em polígonos (SCHULTZ et al., 2017). Desta forma, gera-se uma matriz de distâncias e um grafo associado. Utilizou-se o *software Java OpenStreetMap Editor* (*JOSM*) versão 15.390 para o cadastro dos pontos dos bairros e transportá-los para uma matriz de distâncias em formato de planilha que será manipulado pelo *Microsoft Office Excel* 2013.

A matriz de distâncias é um dado de entrada para a primeira etapa do procedimento heurístico. Outro dado importante são as demandas (pesos) dos arcos, ou seja, o quanto de lixo será coletado em cada rua. Estes dados referentes à quantidade de lixo que será coletado em

cada rua foram determinados seguindo o procedimento abordado por Jacinto, Rosa e Banos (2014) que determinou a média de resíduos sólidos coletados por rua. Para isso, calcula-se o número de residências por rua pela equação 4.1.

$$NDom = \frac{ExtRua}{7{,}5} \tag{4.1}$$

Onde $NDom$ é o número de residências na rua, $ExtRua$ é a extensão (comprimento) de cada rua e 7,5 é um valor fixo adotado da metragem média, em metros, do comprimento de frente de um terreno. Por meio do valor do número de residências por rua, pode-se estimar a quantidade de lixo coletado por rua a partir da equação 4.2.

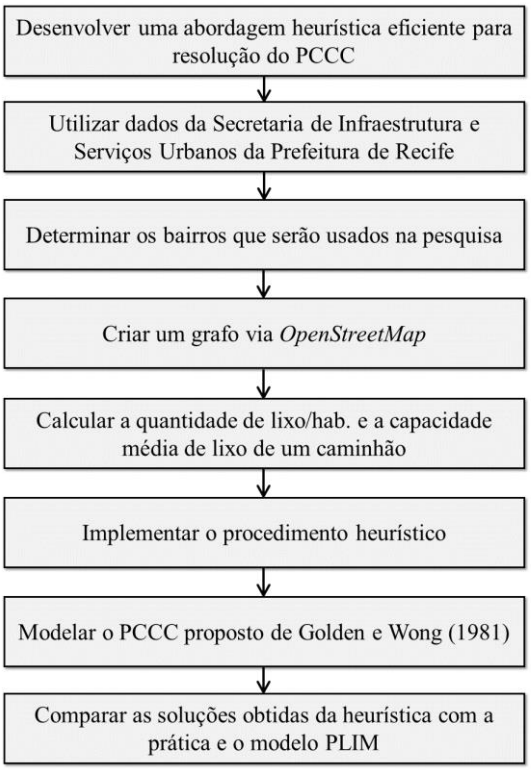
$$Qtlixo = NDom * QLiha * NMha \tag{4.2}$$

Onde $Qtlixo$ representa a quantidade total de lixo gerada por uma rua, $QLiha$ é o número médio de lixo produzido na cidade por habitante por dia e, $NMha$ é a quantidade média de pessoas por residência. Como pode ser observado, a demanda é diretamente proporcional ao comprimento dos arcos (segmentos de ruas). Os dados fornecidos pela Secretaria de Infraestrutura e Serviços Urbanos de Recife e os dados da ABRELPE (2019), fomentaram as demandas para cada rua que será percorrida.

Em seguida, contendo todos os dados de entrada para o algoritmo, então, o modelo traça as rotas viáveis para os caminhões de lixo. O mesmo é feito para o modelo de Golden e Wong (1981). Assim pode-se comparar os dois modelos para verificar a eficiência da heurística abordada com um modelo exato para o PCCC. A Figura 12 apresenta de forma simplificada o passo a passo da metodologia empregada.

Além da aplicação com a coleta de lixo urbano, também foram criadas 3 instâncias artificiais para comparação da heurística com o modelo PLIM de Golden e Wong (1981). As instâncias variam entre |N| = 31 e |A| = 51 e |N| = 315 e |A| = 515.

Figura 12 – Fluxograma para a metodologia deste trabalho



Fonte: O autor (2020)

4.3 DESCRIÇÃO DO PROCEDIMENTO HEURÍSTICO

O procedimento para resolução do PCCC é descrito em quatro etapas. A primeira é a confecção de um programa que escreve um Modelo de Programação Linear (MPL) para o caso direcionado do PCC a partir da matriz de distâncias obtida com auxílio do *JOSM*.

A segunda trata-se da resolução do MPL por um *solver* eficiente, visando encontrar um grafo direcionado balanceado, ou seja, $\forall\, i \in N,\ deg_{in}\,(i) = deg_{out}\,(i)$.

A terceira etapa consiste em encontrar um *tour* Euleriano para o grafo por meio do algoritmo de Hierholzer.

Na quarta e última etapa, visita-se todos os nós de grau maior do que 3 obtidos a partir da etapa 3 e realiza mudanças de transição para encontrar rotas equilibradas se baseando nas capacidades. O Quadro 1 descreve essas etapas em 4 passos, onde é necessário introduzir *inputs* (dados de entrada), descrição do procedimento e *outputs* (dados de saída ou resultado).

Quadro 1 – Procedimento para determinação de m tour

Procedimento do programa para determinação de rotas
Passo 1. **Entrada:** Matriz das distâncias dos segmentos de ruas (comprimento dos arcos). **Processo:** Escrever o Modelo de Programação Linear (MPL) para o PCCD em um arquivo de texto. **Saída:** Arquivo com extensão (.lp) para entrada no *solver*.
Passo 2. **Entrada:** Arquivo com extensão (.lp) para entrada no *solver*. **Processo:** Resolver o MPL com o objetivo de determinar o número de vezes que cada segmento de rua deverá ser percorrido para minimizar a distância total do trajeto do veículo coletor. **Saída:** Quantidade de vezes que cada segmento de rua é percorrido e a distância mínima total.
Passo 3. **Entrada:** Número de vezes que cada segmento de rua é percorrido e a distância mínima. **Processo:** Sequenciar o percurso do caminhão de lixo percorrendo todas as ruas, por meio do algoritmo de Hierholzer modificado. **Saída:** *Tour* Euleriano.
Passo 4. **Entrada:** *Tour* Euleriano. **Processo:** Determinar *V Tours* equilibradas respeitando a capacidade de cada veículo. **Saída:** *Tours* para os *m* veículos coletores.

Fonte: O autor (2020)

4.3.1 Passo 1

O procedimento inicial proposto requer que se resolva o PCCD apresentado na subseção 2.4.4. Para isso, o primeiro passo requer desenvolver um programa simples que escreva o MPL em um arquivo de texto com extensão ".lp" (formato lido pelo *solver SCIP Optimization*) para cada grafo. Embora o modelo descrito na subseção 2.4.4 não seja um MPL, por apresentar variáveis inteiras, mas dada a propriedade de unimodularidade, o mesmo pode ser resolvido como um MPL. Tendo isso em mente, a partir de então será chamado de MPL.

O principal motivo para a criação do programa é que quando se aumenta o tamanho do grafo, então escrever o código de entrada correspondente para o *solver* que será utilizado pode se tornar uma atividade trabalhosa. O programa foi desenvolvido em linguagem de programação *Python* 3.7 e tem como entrada a matriz de distâncias do grafo em formato de planilha ".csv", proveniente do *JOSM*. O MPL em formato ".lp" será utilizado no próximo passo.

4.3.2 Passo 2

A resolução do MPL foi feita pelo *SoPlex*, um *solver* para problemas de programação linear de alta performance presente no *software SCIP Optimization Suite* versão 6.0.1. O *SCIP Optimization* é um dos solucionadores não comerciais mais rápidos para Programação Inteira Mista (PIM) e Programação Não Linear Inteira Mista (PNLIM). O *software* é orientado para as necessidades de especialistas em programação matemática que desejam ter controle total do processo da solução e acessar informações detalhadas até o ponto final do solucionador (ACHTERBERG, 2009; GLEIXNER et al., 2018).

O *SCIP Optimization Suite* possui diversos solucionadores para gerar e resolver diversos tipos de problemas, entre eles destaca-se, o *SCIP*, *solver* de resolução para PIMs e PNLIMs, e o *SoPlex*, *solver* de otimização para resolver MPLs (ACHTERBERG, 2009; GLEIXNER et al., 2018).

O *solver SoPlex* resolve o MPL descrito no passo anterior e encontra o tamanho do percurso total, ou seja, o comprimento total do *tour* de Euler, e a quantidade de vezes que cada arco precisa ser percorrido para que o grafo se torne balanceado. Uma das vantagens do uso do *SoPlex* é a sua rápida execução e resposta, devido a ele ser executado direto no *prompt* de comando de qualquer sistema operacional, assim este resultado pode ser facilmente manipulado por uma linguagem de programação eficiente.

4.3.3 Passo 3

Para determinar o *tour* de Euler utilizou-se o Algoritmo de Hierholzer, conforme apresentado no Algoritmo 2 descrito na subseção 2.4.4. O algoritmo foi implementado em linguagem de programação *Python* 3.7, seguindo o pseudocódigo apresentado a seguir (Figura 13). A saída do algoritmo retorna o *tour* de Euler que será um dos dados de entrada para o próximo passo.

Figura 13 – Algoritmo de Hierholzer para grafos direcionados

Algoritmo de Hierholzer: Modificado para grafos direcionados

```
Entrada:
G = (N, A)                  // Grafo Euleriano
G' = G com G' = (N', A')    // Cópia do grafo original para retornar o tour
T = {∅}                     // Uma lista vazia T para armazenar o tour
n_1 ∈ N'                    // O nó inicial pertence ao grafo G'
   Início:
      T = [n_1]  // O tour inicia no nó inicial
      i = n_1    // O nó i recebe o nó inicial
      Enquanto A ≠ {∅} faça:  // Enquanto o conj. A não for vazio, faça:
         T' = {∅}     // Cria-se um T' para armanezar o tour temporário
         cont = 1     // Um contador que inicia em 1
         Escolher um arco a(i,j) adjacente a i, que não foi percorrido
         T' = T' ∪ j       // Acrecenta o nó rotulado ao tour
         A' = A' − (i,j)   // Retirar o arco a(i,j)∈ A' do grafo
         i = j             // Será atribuído ao nó i o nó da próxima iteração j
         cont = cont + 1    // O contador soma 1
         Enquanto T'[n_1] ≠ T'[n_cont] , faça:
            // Enquanto o 1º nó for diferente do último nó de T', faça:
            Escolher um arco a(i,j) adjacente a i, que ainda não foi percorrido
            T' = T' ∪ j       // Acrecenta o nó rotulado ao tour
            A' = A' − (i,j)   // Retirar o arco a(i,j)∈ A' do grafo
            i = j             // Será atribuído ao nó i o nó da próxima iteração j
            cont = cont + 1 // O contador soma 1
         fim do enquanto
      Concatene T' em T
      fim do enquanto
      Retome T
   Fim
Saída:
T contendo um tour Euleriano
```

Fonte: O autor (2020)

4.3.4 Passo 4

Uma vez apresentado um grafo Euleriano com seu respectivo *tour*, então é acrescentado a esses dados: as distâncias c_{ij} e as demandas d_{ij} de todos os arcos do grafo, a capacidade máxima que cada *tour* pode armazenar (W_T), e por fim, a quantidade de *tours* que se deseja gerar a partir do *tour* original. Esses dados são necessários para qualquer resolução do PCCC.

No passo 4, cada rota gerada é atribuída a um veículo coletor, logo a quantidade de *tours* é igual ao número de veículos (V). A equação 4.3 calcula o limitante inferior para a quantidade de veículos necessários para o procedimento heurístico. O número de veículos, a priori, deve ser maior do que a soma de todas as demandas positivas do grafo dividido pela capacidade

máxima de um veículo. Dado que este valor pode ser um número decimal, então se arredonda para o próximo número inteiro. Este é o principal critério de parada do algoritmo.

$$V = \left\lceil \frac{\sum_{(i,j)\in A} d_{ij}}{W_T} \right\rceil \tag{4.3}$$

A partir dos dados de entrada, o procedimento percorre aleatoriamente os nós que possuem grau de no mínimo 4 e procura mudar uma transição do *tour* para que seja gerado V *tours* disjuntos, em seguida, as rotas são verificadas se atendem a capacidade máxima W_T. O procedimento se repete até que se determinem os *tours* com as distâncias totais mais balanceadas possíveis. Caso a quantidade de rotas não seja suficiente, então acrescenta-se mais um veículo e o procedimento se repete. A descrição minuciosa das características da heurística, do algoritmo proposto e uma aplicação para exemplificar será detalhada no próximo capítulo.

Após o fim do último passo, a solução obtida pela heurística será comparada com a solução obtida pelo modelo de Golden e Wong (1981) que foi implementado no *software SCIP Optimization Suite* versão 6.0.1 no pacote *SCIP* para resolução de modelos PLIM.

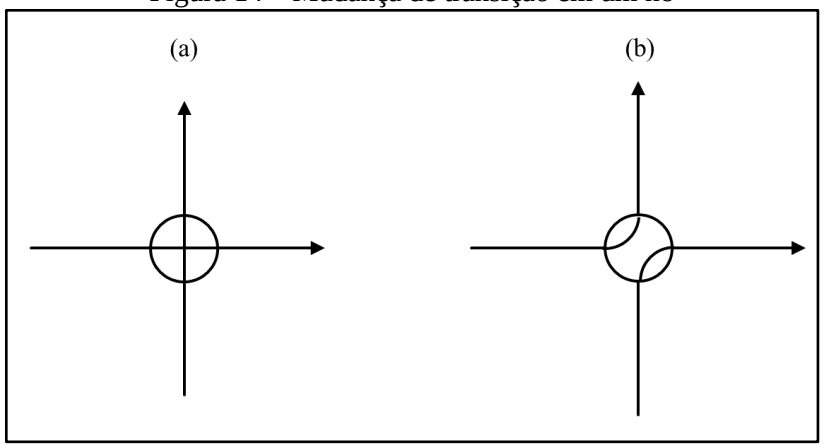
5 HEURÍSTICA PARA QUEBRA DE TOURS (TOUR BREAK HEURISTIC)

Neste capítulo será explicado de forma detalhada as etapas do procedimento heurístico, suas principais características, a descrição do algoritmo e um exemplo em um grafo para sua fixação.

5.1 DESCRIÇÕES PRELIMINARES IMPORTANTES

A elaboração da heurística é baseada no que será chamado a partir daqui como “mudança de transições”. Uma transição é um par de arcos incidentes a um nó, ou seja, um entrando e o outro saindo do nó. Uma mudança de transição em um nó *i* representa um arco que entra em *i* e tem a sua saída alterada para um outro arco que também sai do nó *i*. Na Figura 14, duas transições transformam-se em outras duas.

Figura 14 – Mudança de transição em um nó

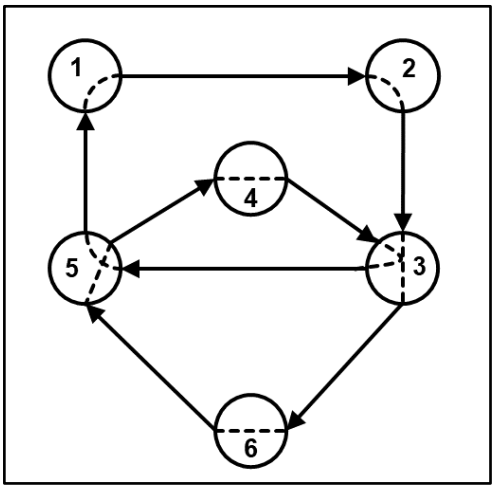


Fonte: O autor (2020)

A Figura 14.a apresenta um nó que é percorrido exatamente 2 vezes para formar um circuito Euleriano. Para verificar a quantidade de transições que este nó possui, é preciso olhar para dentro do nó e verificar que existe apenas uma interseção (ou cruzamento) entre os arcos incidentes a ele, portanto, este nó possui apenas 1 transição. Por apresentar 1 transição, então só é possível ocorrer uma mudança. A Figura 14.b apresenta esta mudança onde um arco incidente agora terá um arco de saída diferente. Tal mudança altera a direção do circuito.

Diante da abordagem apresentada para mudanças de transição em nós, pode-se generalizar para um grafo balanceado. Segue a Figura 15 com um grafo balanceado com um *tour* Euleriano definido: 1 – 2 – 3 – 6 – 5 – 4 – 3 – 5 – 1.

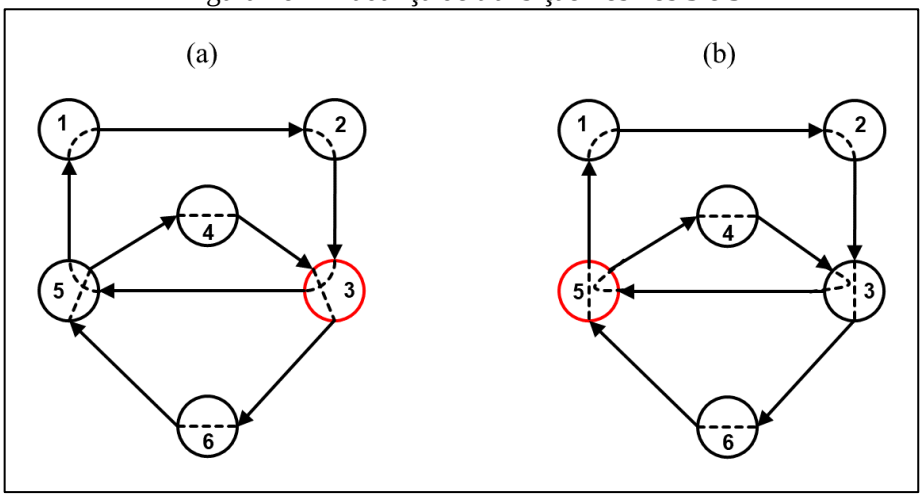
Figura 15 – Grafo com um Tour Euleriano bem definido



Fonte: O autor (2020)

Na Figura 15, é possível verificar que os nós 3 e 5 apresentam grau 4, portanto, pode-se executar uma mudança de transição em cada nó. A Figura 16 apresenta as duas mudanças de forma separada, onde percebe-se que tal “quebra” muda a direção do circuito.

Figura 16 – Mudança de transição nos nós 3 e 5



Fonte: O autor (2020)

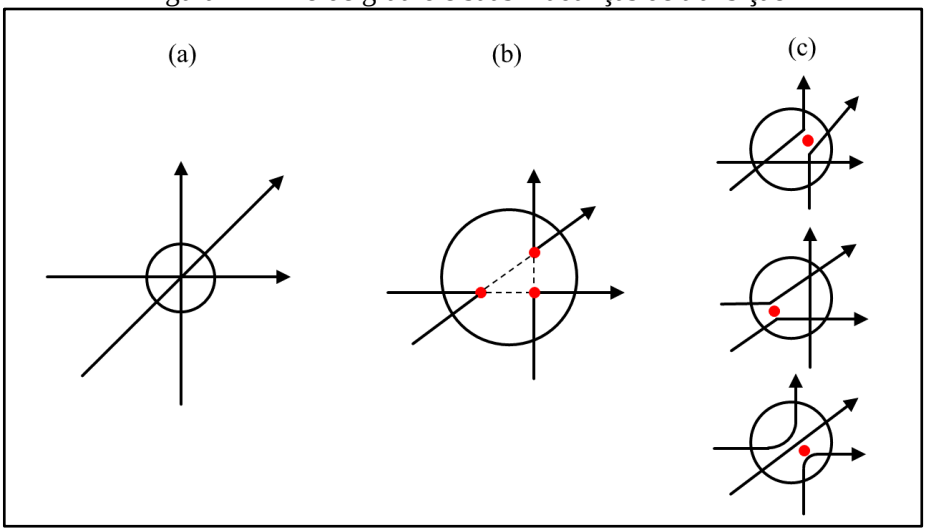
Primeiramente, analisando a Figura 16.a, a mudança de transições no nó 3 mudou a direção dos arcos em seu caminho. É possível perceber que a mudança de transição gerou dois circuitos fechados independentes, como segue: 1 – 2 – 3 – 5 – 1 e 3 – 6 – 5 – 4 – 3. Será chamado o primeiro circuito fechado de rota 1 e o segundo de rota 2. Para a Figura 16.b a mudança de transição no nó 5 também criou dois circuitos disjuntos: 1 – 2 – 3 – 6 – 5 – 1 e 3 – 4 – 5 – 3. Portanto, conclui-se que ao mudar a transição de um nó com grau maior do que 2 em um grafo

dirigido com um circuito Euleriano, ocorre uma “quebra” que, por sua vez, criam-se dois circuitos disjuntos.

O resultado apresentado acima é importante quando se precisa quebrar um *tour* Euleriano em dois *tours* disjuntos. O que é o caso de muitas aplicações práticas. Porém, a Figura 16 apresentou duas quebras diferentes, que no que lhe concerne, apresentaram circuitos diferentes, então como saber qual a melhor solução? À vista disso, é necessário analisar as capacidades dos arcos. Quando as capacidades dos arcos do grafo da Figura 16 forem considerados, então será possível decidir se a melhor mudança de transição ocorrerá no nó 3 ou nó 5.

Diante do discutido para um nó com grau 4, espera-se que se distinga o caso quando um nó conter grau superior a 4. Para isso, a Figura 17.a apresenta um nó de grau 6, onde se determina que a mudança de transição pode ocorrer de 3 maneiras diferentes (Figura 17.b). Vale salientar, que aqui serão sempre feitas mudanças de transição trocando apenas a direção de dois arcos entrando e dois saindo, mesmo que o nó apresente grau superior à 4. Assim, serão realizadas sempre uma mudança de transição por vez, e nunca duas ou mais simultaneamente, devido este número crescer exponencialmente com o aumento do grau do nó.

Figura 17 – Nó de grau 6 e suas mudanças de transição



Fonte: O autor (2020)

Os 3 pontos em destaque na Figura 17.b permitem visualizar que para o caso de grau 6, a mudança de transição pode ocorrer de 3 formas diferentes (Figura 17.c), sendo que em cada caso é gerado dois *tours* disjuntos. Para cada mudança de transição é necessário mostrar as duas rotas geradas para se definir qual apresenta a melhor solução. O mesmo pode ser feito quando se aumenta o grau de um nó qualquer. A Tabela 1 apresenta a quantidade de mudanças de

transições possíveis quando se aumenta o número de arcos incidentes a um nó em um grafo dirigido. Observa-se que o número de mudanças cresce com o aumento do grau do nó.

Tabela 1 – Quantidade de mudanças de transição com o aumento do grau de um nó

Grau do nó (*i*)	**2**	**4**	**6**	**8**	**10**	**12**	**Fórmula**
Quantidade de mudanças de transições possíveis (*n*)	0	1	3	6	10	15	$n_i = n_{i-2} + ({}^{i}/_{2} - 1)$

Fonte: O autor (2020).

Determinar qual quebra gera a melhor solução deve levar em conta qual o objetivo deseja ser atingido. Neste caso, será considerado como objetivo determinar rotas com comprimentos totais equilibrados, ou seja, que apresentem a soma das distâncias de cada rota mais aproximadas possível. Então deve-se considerar o comprimento de todo arco do grafo.

Além de rotas balanceadas, cada rota gerada deve atender a capacidade máxima conhecida (a capacidade de lixo a ser coletado), então a demanda total de cada rota não deverá ultrapassar a capacidade máxima de um *tour*. Essa restrição entra como uma decisão ao ser encontrada uma rota viável. Porém, quando uma rota não atender ao perfil da restrição de capacidade total a mesma não é excluída, a priori. Essa rota poderá ser quebrada em mais dois *tours* distintos e verificada se suas duas novas rotas são as mais balanceadas e se é uma solução factível para o problema. Caso contrário, aquele nó não gera uma boa quebra de rotas e procura-se outro nó viável.

Na próxima subseção será descrito o pseudocódigo do algoritmo heurístico e denominado as etapas do procedimento.

5.2 DESCRIÇÃO DO PROCEDIMENTO HEURÍSTICO

A abordagem heurística proposta segue 4 etapas bem definidas de busca e decisão em um grafo balanceado com suas distâncias e demandas conhecidas. O algoritmo heurístico é apresentado na Figura 18 e por se tratar de um procedimento que busca “quebrar” um *tour* para *V* rotas, este será chamado de *Tour Break Heuristic* (Heurística de Quebra de *Tour*) ou *TourBreakHeuristic*.

Figura 18 – Algoritmo Tour Break Heuristic

Algoritmo: Tour Break Heuristic

1) **Entrada: Etapa 1**
2) $G = (N, A)$ // Recebe um Grafo Balanceado
3) $\boldsymbol{T} \in \boldsymbol{G}$ // *Tour* Euleriano
4) W_T = Capacidade máxima de cada veículo
5) $V = \left\lceil \sum_{(i,j)\in A} d_{ij}/W_T \right\rceil$ // Quantidade de veículos necessários
6) *TourA* = [∅] // Lista vazia para receber a sequência de arcos (i, j) do *Tour*
7) **Início:**
8) **Etapa 2**
9) **Para todo** nó de $\boldsymbol{T}$, **enumere** o arco (i, j) correspondente e **adicione** em *Tour*A
10) *double_vertex* = [] // Lista para armazenar os nós de grau maior do que 3
11) **Para todo** nó $i \in T$:
12) **Se** $|T(n_i)| > 3$, **faça:** // Se o grau de i for maior que 3
13) double_vertex = double_vertex $\cup\, n_i$
14) **Etapa 3**
15) // Criar listas e variáveis
16) **Para todo** v em V, **faça:**
17) $R_v = Rota_v = [\emptyset]$ // Listas vazias
18) $Dist_v = Dist\_T_v = W_v = WT_v = 0$ // Variáveis iniciam em 0
19) *Dif_D_Global* = $\boldsymbol{M}$ e *Dif_D_Atual* = |$\boldsymbol{M}$-1| // Parâmetros de comparação
20) **Enquanto** ***double_vertex*** $\neq \emptyset$, **faça:** // Enquanto a lista não for vazia, faça:
21) x = um nó **aleatório** de double_vertex
22) double_vertex = double_vertex - x
23) **Para todo** arco (i, j) adjacente a x, **faça:**
24) Insira (i, j) em *Ligacoes_de_x*
25) **Para toda** *transição* em *Ligacoes_de_x*, **faça:**
26) **Mudar** a transição
27) **Para todo** v em V, **faça:**
28) R_v: recebe a rota do v-ésimo circuito fechado
29) **Para todo** v em R_v, **faça:**
30) W_v: recebe a Demanda Total da v-ésima rota
31) **Etapa 4**
32) **Se** $W_v \leq W_T$, $\forall\, v \in V$, **faça:** // Se as demandas atendem a Capacidade
33) **Para todo** v em R_v, **faça:**
34) $Dist_v$: recebe a Distância Total do v-ésimo circuito
35) *Dif_D_Atual* = $\frac{\sum_{v}^{V-1} |Dist_v - Dist_{v+1}|}{V}$
36) **Senão:** // Se pelo menos uma demanda não atende a capacidade
37) *Dif_D_Atual* = $\boldsymbol{M}$
38) **Se** *Dif_D_Atual* < *Dif_D_Global*, **faça:**
39) *Dif_D_Global* = *Dif_D_Atual*
40) **Para todo** v em V, **faça:**
41) $Rota_v = R_v$
42) $Dist\_T_v = Dist_v$
43) $WT_v = W_v$
44) Melhor_No = x
45) **fim do enquanto**
46) **Se** nenhuma Quebra Viável for encontrada, **faça:**
47) $V = V + 1$
48) **Retorne a Etapa 3**
49) **Fim**
50) **Saída:**
51) Rotas disjuntas e suas respectivas Distâncias e Demandas totais

Fonte: O autor (2020)

O algoritmo recebe o grafo balanceado (linha 2) no início da etapa 1, assim como o *Tour* Euleriano de *G* (linha 3) e a capacidade máxima permitida por cada veículo v (linha 4). Na linha 5 é calculado a quantidade de veículos necessários para iniciar o procedimento de "quebra de *tour*". Esse é o principal critério de parada do algoritmo, que se tornou essencial para evitar que o algoritmo continue quebrando *tours*, quando já exista uma solução viável com menos rotas. Na linha 6, cria-se uma lista vazia (*TourA*), que irá receber os arcos (*i, j*) do *Tour*.

Na etapa 2, o *TourA* é preenchido com a sequência de nós adjacentes (*i, j*) de G presentes no *Tour*. A linha 10 apresenta a criação de uma lista (*double_vertex*) que irá receber apenas os nós cuja mudança de transição é possível de ser realizada, ou seja, nós de grau superior a 3. As linhas 11-13 verificam então qual(is) o(s) nó(s) do *Tour*, que podem sofrer mudança de transição e o(s) insere(m) na lista *double_vertex*.

A etapa 3 inicia criando novas listas e variáveis de acordo com a quantidade de veículos (linhas 16-18): R_v recebe as rotas atuais de cada nó após a mudança de transição, enquanto que as melhores rotas encontradas no final de cada iteração são atualizadas em $Rota_v$. W_v armazena a demanda total de cada rota gerada, assim como, $Dist_v$ guarda a distância total de cada rota atual. $Dist\_T_v$ e WT_v são atualizadas com as distâncias totais e as demandas totais, respectivamente, das melhores rotas determinadas ao final de cada iteração.

Para verificar se as rotas atuais são mais equilibradas do que outras, criam-se variáveis de comparação (linha 19). *Dif_D_Atual* é a diferença média absoluta entre as distâncias de cada rota, onde seu valor indica o quanto as rotas estão balanceadas, onde 0 representa rotas iguais. A equação 5,1 apresenta o cálculo para *Dif_D_Atual*. Inicialmente esse parâmetro de comparação recebe um número tão grande, porém menor do que $\boldsymbol{M}$. Já *Dif_D_Global* receberá *Dif_D_Atual* se as rotas atuais forem mais balanceadas do que a anterior. *Dif_D_Global* inicia com um $\boldsymbol{M}$ grande, o que força a primeira mudança de transição escolhida a ser aceita, mesmo que ela não seja a mais viável.

$$Dif\_D\_Atual = \frac{\sum_{v}^{V-1} |Dist_v - Dist_{v+1}|}{V} \tag{5.1}$$

O "*loop* enquanto", presente na linha 20, permite testar todos os nós de *double_vertex*, um por vez, até que todos tenham sido testados. A escolha dos nós é feita de forma aleatória (linha 21) e em seguida a lista remove o nó selecionado *x* (linha 22). Na linha 23 é selecionado todos os arcos que entram e saem do nó *x* e são inseridos em uma lista (*Ligacoes_de_x*) em ordem, conforme linha 24. A ordem dos arcos se faz necessário, para a próxima ação. Cada transição é escolhida por vez (linha 25) para realizar a mudança de transição (linha 26), como já mostrado na seção 5.1.

Para cada mudança de transição de *Ligacoes_de_x*, cada rota é inserida nas variáveis $R_1, R_2, \dots, R_V$ por meio do *v*-ésimo circuito gerado pela mudança de transição (linhas 27-28). De forma análoga, para cada rota gerada calcula-se a demanda total que é armazenada nas variáveis $W_1, W_2, \dots, W_V$ (linhas 29-30).

A etapa 4 inicia com uma decisão importante, que respeita a capacidade máxima de cada veículo (linha 32), logo se todas as demandas W_v atenderem ao quesito, então para toda rota gerada calcula-se a distância total e insere os valores nas variáveis $Dist_1, Dist_2, \dots, Dist_V$ (linhas 33-34). Na linha 35 é calculado a diferença absoluta das somas das distâncias entre as V rotas atuais que é armazenada na variável *Dif_D_Atual*.

Se a decisão da linha 32 for verdadeira, então é verificado se *Dif_D_Atual* é menor do que a diferença das distâncias assumida como a melhor (*Dif_D_Global)* (linha 38).

A linha 38 sendo verdadeira, então as rotas são viáveis e as variáveis de melhor quebra são atualizadas (linhas 39-44). A variável *Dif_D_Global* é atualizada (linha 39). Para cada rota gerada será feito: $Rota_v$ recebe as rotas dos *V tours* gerados (linha 41), $Dist\_T_v$ recebe as distâncias referentes as rotas (linha 42) e WT_v armazena as demandas oriundos de W_v (linha 43). Por fim, o nó que realizou a melhor mudança de transição é armazenado na variável *Melhor_No* (linha 44).

Caso a decisão na linha 38 seja falsa, então descarta-se a mudança de transição e parte para uma nova transição no nó *x*, caso ainda existam transições a serem testadas (linha 25), ou escolhe-se outro *x* no início da etapa 3. Caso a decisão da linha 32 não seja verdadeira, então significa que aquela mudança de transição não gerou boas rotas, logo a variável *Dif_D_Atual* irá receber um ***M*** grande (linha 37) para não entrar na próxima decisão (linha 38) e retorna para a linha 25.

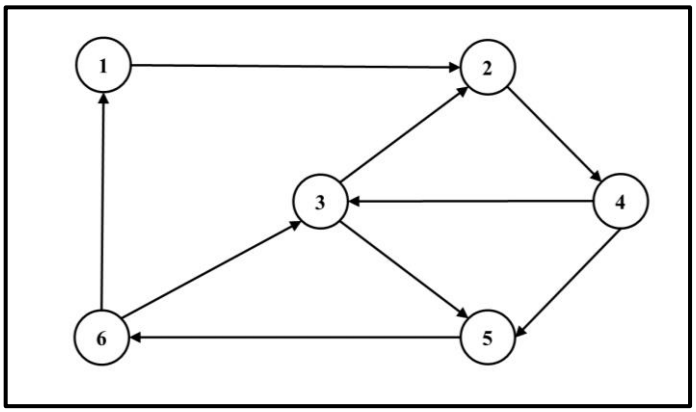
Caso o procedimento não encontre nenhuma rota viável com nenhuma quebra de *tour* (linha 46), então deve-se aumentar um veículo em V (linha 47) e retornar a etapa 3 novamente (linha 48). No final, o procedimento retorna as V rotas viáveis com cada distância e demanda totais associadas.

Para melhor entendimento do algoritmo, ele será usado em um exemplo simples para um grafo não balanceado.

5.3 APLICAÇÃO DO PROCEDIMENTO HEURÍSTICO PROPOSTO

O grafo modelo para aplicação do *TourBreakHeuristic* é o grafo direcionado *M*, que foi utilizado como motivação para o procedimento heurístico proposto. O grafo contém 6 nós, 9 arcos, é não balanceado, e está representado na Figura 19. A Tabela 2 apresenta os dados referentes a distância e a demanda de cada arco. Será considerado: W_T = 23 unidades (capacidade máxima).

Figura 19 – Grafo não Euleriano M



Fonte: O autor (2020).

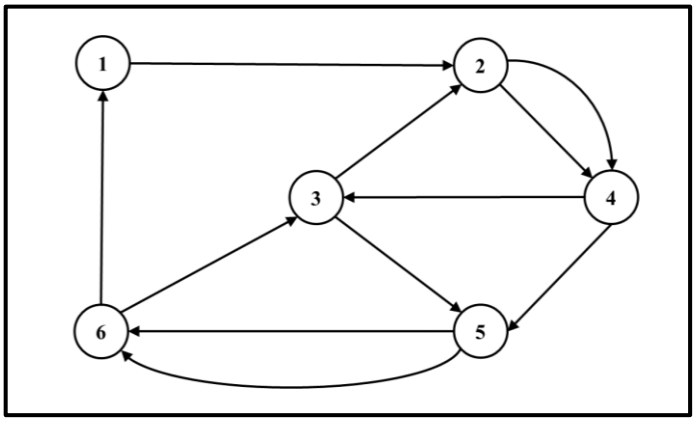
Tabela 2 – Distâncias e demandas para o grafo M

Dados	**Arcos do grafo *M***								
	(1, 2)	**(2, 4)**	**(3, 2)**	**(3, 5)**	**(4, 3)**	**(4, 5)**	**(5, 6)**	**(6, 1)**	**(6, 3)**
c_{ij}	5	6	10	5	7	7	8	10	6
d_{ij}	3	3	5	3	4	4	4	5	3

Fonte: O autor (2020)

O primeiro passo do procedimento é a escrita do Modelo de Programação Linear (MPL) para resolução pelo *solver SoPlex* (passo 2), que retornará um grafo balanceado. Assim, a resposta do programa retorna a quantidade de vezes que cada arco é atravessado, como apresenta a Figura 20 com o grafo já balanceado, onde a Função Objetivo apresentou o valor de 78, ou seja, $\sum c_{ij}$ = 78. No próximo passo é determinado o *Tour* Euleriano pelo algoritmo de Hierholzer (passo 3). Sua resolução apresentou a seguinte sequência de nós do *tour*: 1 – 2 – 4 – 3 – 2 – 4 – 5 – 6 – 3 – 5 – 6 – 1.

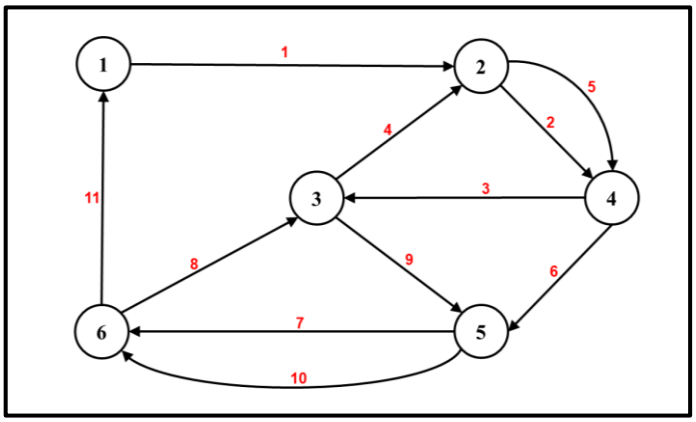
Figura 20 – Grafo M balanceado



Fonte: O autor (2020)

O último passo é a aplicação do algoritmo heurístico, onde é definido a priori que cada *tour* não pode exceder a demanda total de 23 unidades. Dado isso, como a soma das demandas do grafo é igual a 34, determina-se $V = 2$, o número de veículos (rotas) mínimo para cobrir todos os arcos A etapa 2 se inicia rotulando a sequência de arcos em ordem crescente conforme são percorridos pelo *tour* traçado no passo 3. A sequência está presente na Figura 21. Em seguida, se determina quais são os nós que possuem grau maior do que 3 para entrar na lista *double_vertex*. O que facilmente pode ser verificado quais destes nós apresentam essa característica, como segue: *double_vertex* = [2, 3, 4, 5, 6].

Figura 21 – Grafo M balanceado e com a sequência de arcos do tour

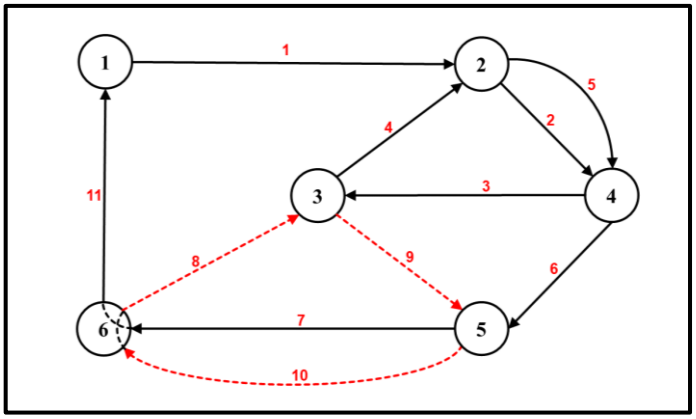


Fonte: O autor (2020)

Escolhendo um nó arbitrário para ser marcado, tem-se o nó 6. O nó 6 apresenta grau 4, logo só possui 1 mudança de transição, que seria mudar a saída do arco 7 para o arco 11 e o arco 10 para o arco 8, onde a Figura 22 apresenta esta mudança e diferencia os dois *tours* em um circuito com linha contínua (rota 1) e outro em linha tracejada (rota 2). Cada *tour* calcula

seu peso total e insere nas variáveis: W_1 = 28 e W_2= 6. No início da etapa 4 se verifica que o peso total W_1 ultrapassou a capacidade máxima de 23, logo a rota 1 ultrapassou a capacidade, então exclui-se a possibilidade dessa mudança de transição. Como não há mais mudanças de transição possíveis em 6, então o próximo nó escolhido é o 5 para gerar duas rotas viáveis.

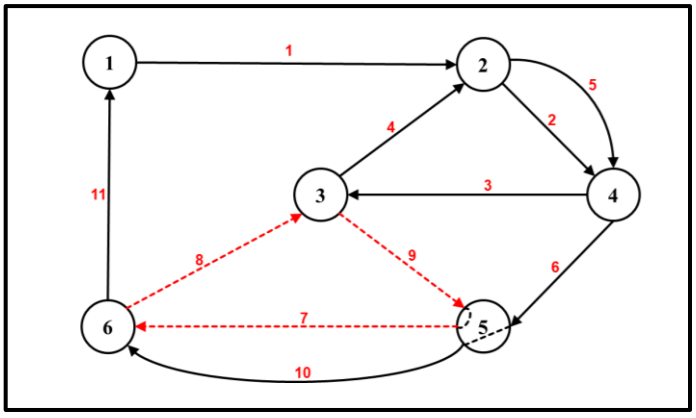
Figura 22 – Mudança de transição no nó 6



Fonte: O autor (2020)

O nó 5 possui grau 4 e sua mudança de transição pode gerar somente a rota 1 = [10, 11, 1, 2, 3, 4, 5, 6] e a rota 2 = [7, 8, 9]. O cálculo dos pesos apresenta: W_1 = 24 e W_2 = 10. Como W_1 ultrapassa a capacidade máxima, então, descarta-se a quebra no nó 5 para gerar 2 rotas viáveis. A Figura 23 apresenta o grafo com as duas rotas da mudança de transição no nó 5.

Figura 23 – Mudança de transição no nó 5

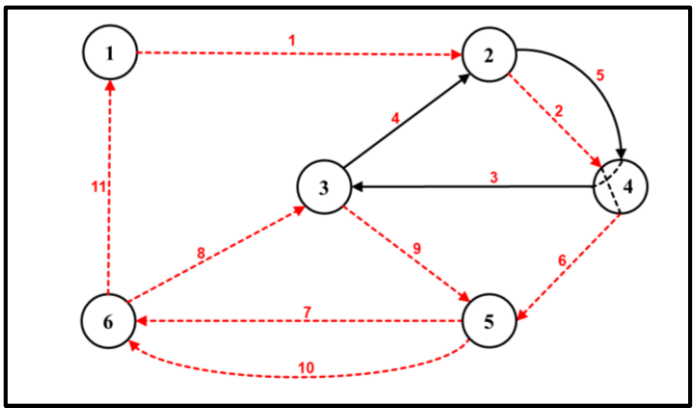


Fonte: O autor (2020)

O próximo nó escolhido é o nó 4. Que apresenta os *indegrees*: 2 e 5 e *outdegrees*: 3 e 6. A sua única mudança de transição gera as rotas rota 1 = [6, 7, 8, 9, 10, 11, 1, 2] com W_1 = 25 e

rota 2 = [3, 4, 5] com W_2 = 9. Devido a demanda total da primeira rota ultrapassar a capacidade máxima, então descarta-se o nó 4 de uma possível quebra para duas rotas viáveis. A Figura 24 mostra as rotas encontradas na mudança de transição para o nó 4.

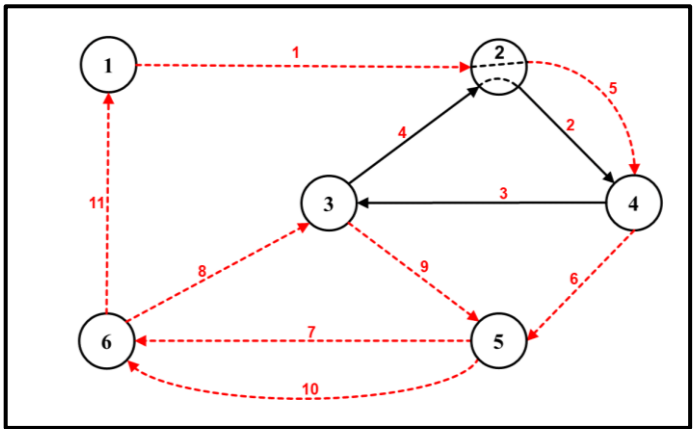
Figura 24 – Mudança de transição no nó 4



Fonte: O autor (2020)

O nó 2 é o próximo escolhido, que apresenta apenas uma mudança de transição do arco 1 para o arco 5 e do arco 4 para o arco 2, como pode ser visto na Figura 25. As rotas obtidas são: rota 1 = [5, 6, 7, 8, 9, 10, 11, 1] e rota 2 = [2, 3, 4]. Calculado os pesos totais W_1 e W_2, obtém-se os valores, 22 e 12, respectivamente. Como nenhuma demanda ultrapassou a capacidade máxima, então, calcula-se a distância total para cada rota ($Dist_1 = 55$ e $Dist_2 = 23$). O valor de *Dif_D_Atual* calculado é $(|55 - 23|/2) = 16$. Como 16 é menor do que um ***M*** grande, então a quebra em 2 assume a melhor quebra até então.

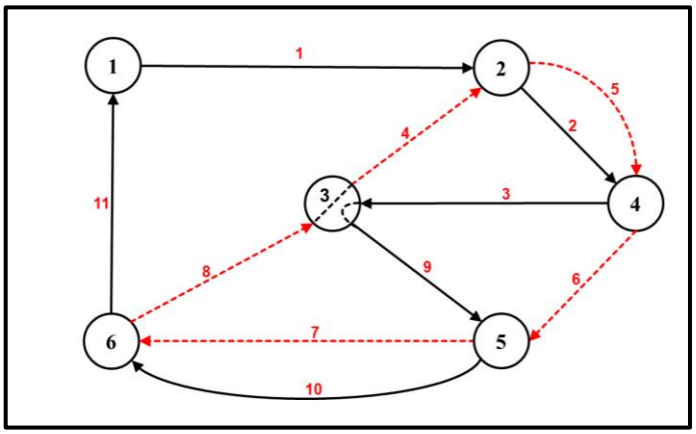
Figura 25 – Mudança de transição no nó 2



Fonte: O autor (2020)

Por fim, o nó 3 é escolhido. Sua mudança de transição muda a direção do arco 3 para o 9 e do arco 8 para o arco 4. Encontra-se para a rota 1: R_1 = [9, 10, 11, 1, 2, 3] e seu peso total W_1 = 18, de forma análoga para a rota 2: R_2 = [4, 5, 6, 7, 8] e W_2 = 16. Entre os pesos totais, nenhum ultrapassou a capacidade máxima. Logo, calcula-se a distância total para cada rota: $Dist_1 = 41$ e $Dist_2 = 37$. A diferença absoluta entre as distâncias (*Dif_D_Atual*) é igual a 2. A Figura 26 apresenta as rotas para a quebra no nó 3.

Figura 26 – Mudança de transição no nó 3



Fonte: O autor (2020)

A próxima decisão consiste em verificar se a diferença absoluta entre as distâncias (*Dif_D_Atual*) é menor do que a diferença absoluta das distâncias assumida no nó 2 (*Dif_D_Global*) que é 16. Como 2 é menor do que 16, então, assume que a quebra em 3 é melhor do que a quebra em 2. Por não apresentar mais nós a serem marcados, então a melhor quebra ocorre no nó 3.

Por fim, concluído a heurística, o próximo capítulo aborda os principais resultados acerca de sua implementação e aplicação em dois bairros da cidade de Recife/PE. O algoritmo também foi implementado para avaliar três instâncias e seus parâmetros de entrada.

6 RESULTADOS E DISCUSSÕES

O capítulo 6 apresenta a aplicação da heurística proposta para o problema da coleta de lixo urbano na cidade de Recife-PE em dois bairros da cidade e sua devida discussão acerca do assunto. Em seguida, três instâncias são apresentadas para resolução do PCCC por meio da heurística e do modelo PLIM de Golden e Wong (1981), onde são analisadas algumas variações de parâmetros para resolução do problema e calculado *gaps* para a solução encontrada.

6.1 APLICAÇÃO PARA A COLETA DE LIXO URBANO EM DOIS BAIRROS

O bairro Engenho do Meio é um bairro de pequeno porte contendo cerca de 45 ruas, cuja grande maioria são compostas de segmentos de ruas em um único sentido de tráfego e compreendido por casas residenciais e pequenos comércios. Já o bairro Cordeiro é um bairro de médio porte apresentando cerca de 86 ruas, também com a grande maioria de seus segmentos de ruas em um único sentido de tráfego e contendo prédios residenciais, casas e pequenos comércios e um hospital de grande porte.

Segundo os dados coletados, a coleta na cidade ocorre por setores definidos em planejamento, onde cada veículo coletor é responsável por um setor com capacidade variando com a demanda do bairro. O atendimento no bairro Engenho do Meio contempla dois setores e o atendimento no bairro Cordeiro apresenta três setores, ou seja, não se realiza a coleta completa para cada bairro por um único veículo coletor.

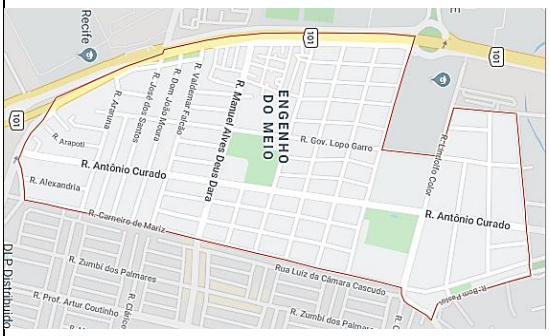
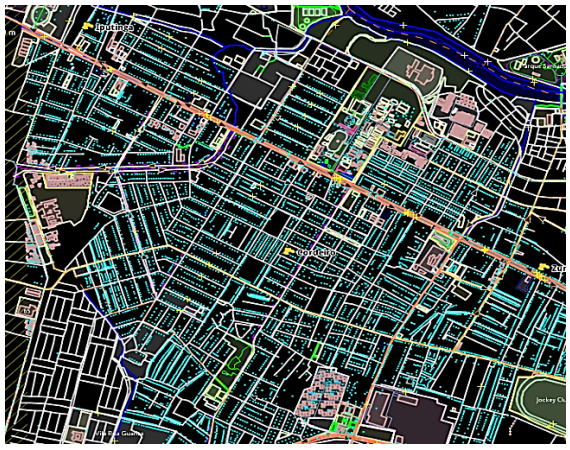
As coletas são feitas em dias alternados com horários bem definidos. Não se teve ao certo o tamanho da distância total percorrida por cada veículo, mas sabe-se que a distância total de cada rota permanece em torno 15 km (quilômetros) para cada setor no bairro Engenho do Meio e de 30 km para cada setor no bairro Cordeiro. Esses dados são para cada setor, uma vez que o veículo sai da garagem, coleta o lixo no bairro e deposita o lixo no aterro sanitário da cidade. O veículo coletor de lixo que é utilizado possui uma capacidade máxima de lixo compensado de 15 m^3, ou seja, aproximadamente 6.500 kg de lixo.

Outro dado interessante é a quantidade de lixo produzido por habitante por dia que chega a aproximados 0,91 kg. Com esses dados coletados para a entrada da resolução do PCCC, geram-se os pesos para cada segmento de rua percorrido que são iguais ao $Qtlixo$ (quantidade total de lixo gerada por um segmento de rua) pela equação 4.2.

Os bairros são organizados e apresentados na Figura 27, onde se tem uma visão da delimitação de cada bairro no *Google Maps* (à esquerda) e o mapeamento dos dados no *Java*

OpenStreetMap Editor (*JOSM*) (à direita) para a geração dos grafos com saída em planilha de dados.

Figura 27 – Bairros mapeados pela pesquisa no JOSM

Bairro Engenho do Meio	**Representação no *JOSM***
	
Bairro Cordeiro	**Representação no *JOSM***
	

Fonte: O autor (2020)

O bairro Engenho do Meio gerou um grafo com $|N| = 116$ nós e $|A| = 202$ arcos, enquanto que o bairro Cordeiro apresentou um grafo com $|N| = 330$ nós e $|A| = 605$ arcos. Estes dados foram resolvidos de forma exata para o PCC direcionado e geraram as soluções apresentados no Quadro 2. O valor para o tempo de resolução está em segundos e inclui o tempo de gerar o arquivo ".lp" e resolvê-lo. O valor da função objetivo está em metros. O resultado do PCCD que tornou os dois grafos balanceados apresentou um aumento natural, já esperado, do número de arcos, logo, o grafo para o Engenho do Meio apresentou após a resolução 269 arcos, ao mesmo tempo que o grafo para o Cordeiro resultou em 994 arcos.

Quadro 2 – Resultados do SoPlex para resolução dos bairros do estudo

Engenho do Meio	**Cordeiro**
SoPlex status: problem is solved [optimal] Solving time (sec): 0.2954 Iterations: 89 Objective value: 23.337	SoPlex status: problem is solved [optimal] Solving time (sec): 0.4221 Iterations: 342 Objective value: 74.200

Fonte: O autor (2020)

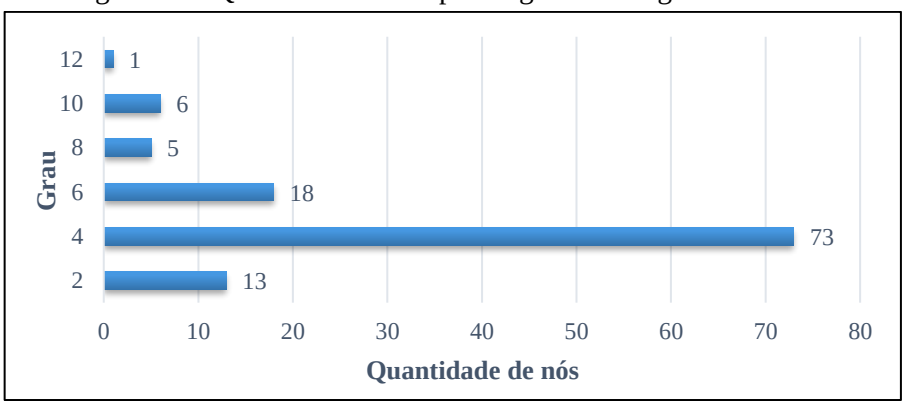
A determinação do *tour* Euleriano no passo 3 do procedimento representou um único percurso para o caminhão de lixo. O Quadro 3 apresenta o *tour* para o bairro Engenho do Meio e o Quadro 4 para o bairro Cordeiro. Já as Figuras 28 e 29 descrevem para os bairros Engenho do Meio e Cordeiro, respectivamente, a quantidade de nós de grau 2, 4, 6, 8, etc. Estas Figuras tornam mais perceptível o quanto o grafo tende a aumentar o número de vezes que cada nó é visitado quando o grafo se torna balanceado.

Quadro 3 – Tour Euleriano para o bairro Engenho do Meio

Engenho do Meio
[1, 2, 113, 114, 115, 10, 115, 116, 13, 12, 11, 23, 35, 47, 48, 49, 50, 51, 52, 53, 62, 75, 77, 83, 77, 75, 62, 53, 46, 47, 52, 63, 68, 67, 66, 76, 77, 75, 62, 61, 112, 69, 75, 62, 53, 46, 34, 22, 34, 46, 53, 46, 47, 48, 36, 24, 12, 11, 10, 115, 116, 13, 25, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 14, 2, 113, 114, 7, 6, 5, 17, 29, 41, 42, 30, 18, 6, 5, 4, 3, 15, 16, 17, 18, 19, 20, 21, 22, 10, 22, 10, 9, 8, 7, 19, 31, 43, 44, 45, 46, 34, 22, 10, 9, 21, 33, 45, 46, 34, 22, 23, 24, 25, 37, 49, 50, 65, 64, 63, 62, 61, 60, 59, 58, 57, 108, 73, 72, 109, 58, 57, 56, 55, 54, 38, 26, 14, 15, 27, 39, 40, 41, 42, 43, 44, 32, 20, 8, 7, 19, 31, 43, 59, 110, 71, 70, 69, 75, 68, 67, 64, 51, 48, 49, 50, 65, 66, 76, 84, 85, 86, 83, 84, 85, 93, 85, 93, 104, 102, 101, 100, 91, 86, 83, 77, 78, 79, 74, 73, 72, 71, 70, 111, 60, 59, 110, 71, 78, 82, 87, 88, 89, 97, 89, 90, 99, 90, 91, 86, 91, 92, 93, 104, 102, 103, 102, 103, 100, 91, 92, 101, 100, 91, 100, 96, 98, 96, 95, 80, 81, 82, 87, 88, 94, 81, 82, 83, 86, 87, 88, 94, 95, 80, 79, 74, 105, 106, 107, 54, 38, 39, 40, 28, 16, 4, 3, 2, 1]

Fonte: O autor (2020)

Figura 28 – Quantidade de nós para o grafo do Engenho do Meio



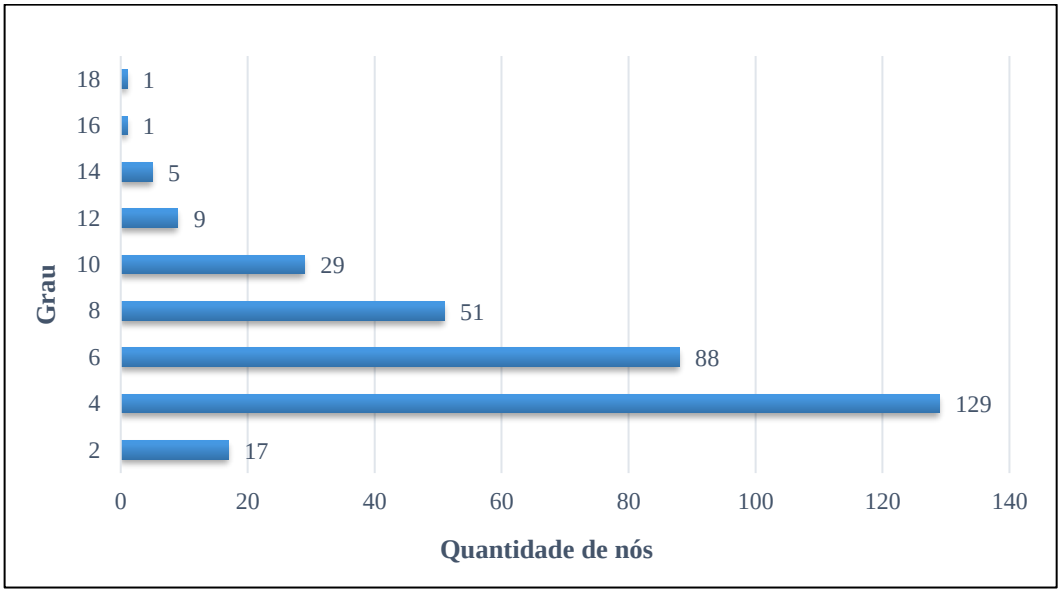
Fonte: O autor (2020)

Quadro 4 – Tour Euleriano para o bairro Cordeiro

Cordeiro
[1, 2, 3, 15, 19, 20, 16, 15, 19, 20, 21, 22, 17, 4, 17, 22, 27, 30, 29, 28, 23, 24, 29, 28, 23, 18, 19, 24, 25, 20, 21, 26, 27, 30, 29, 28, 23, 18, 14, 2, 3, 4, 5, 6, 72, 69, 68, 70, 71, 6, 7, 106, 105, 104, 76, 79, 109, 107, 108, 107, 103, 104, 76, 77, 78, 67, 66, 68, 70, 71, 6, 7, 106, 75, 76, 77, 74, 73, 78, 67, 66, 68, 70, 71, 6, 7, 106, 75, 74, 73, 72, 69, 67, 65, 62, 60, 59, 40, 31, 40, 39, 32, 31, 40, 39, 38, 37, 36, 35, 28, 23, 24, 25, 26, 27, 30, 31, 40, 47, 40, 47, 46, 45, 44, 41, 36, 35, 34, 33, 38, 43, 46, 45, 44, 41, 42, 43, 46, 45, 48, 49, 51, 53, 54, 56, 157, 158, 161, 162, 165, 164, 161, 162, 163, 158, 163, 158, 167, 168, 169, 170, 171, 172, 52, 172, 173, 172, 173, 174, 175, 166, 163, 166, 165, 164, 161, 162, 163, 166, 177, 166, 177, 176, 191, 190, 177, 190, 319, 178, 164, 161, 162, 163, 170, 171, 172, 173, 180, 173, 180, 181, 174, 175, 176, 191, 190, 319, 178, 177, 190, 319, 178, 179, 160, 159, 156, 126, 128, 136, 147, 326, 325, 324, 310, 311, 310, 324, 323, 322, 211, 311, 312, 311, 312, 313, 314, 315, 316, 317, 318, 317, 318, 193, 318, 193, 194, 195, 200, 212, 213, 218, 220, 224, 227, 234, 235, 293, 235, 234, 233, 234, 233, 237, 243, 255, 256, 259, 262, 285, 286, 309, 186, 307, 308, 309, 186, 147, 148, 149, 150, 151, 150, 151, 152, 151, 152, 153, 152, 153, 154, 153, 154, 320, 321, 330, 329, 150, 155, 149, 150, 155, 151, 330, 329, 328, 327, 148, 149, 328, 327, 326, 147, 148, 147, 136, 137, 146, 145, 311, 145, 146, 147, 146, 232, 325, 324, 323, 322, 310, 232, 325, 324, 310, 311, 145, 144, 143, 313, 314, 142, 140, 141, 315, 314, 313, 314, 315, 316, 315, 316, 317, 316, 317, 318, 193, 194, 195, 196, 197, 194, 195, 200, 212, 213, 214, 215, 217, 240, 217, 245, 246, 247, 252, 254, 247, 252, 187, 269, 270, 272, 273, 271, 270, 272, 273, 274, 275, 276, 277, 278, 279, 282, 277, 278, 279, 280, 308, 309, 186, 147, 136, 128, 126, 127, 129, 130, 136, 128, 127, 129, 130, 136, 128, 126, 87, 126, 87, 86, 85, 57, 58, 59, 40, 47, 55, 49, 51, 50, 48, 50, 52, 172, 173, 180, 181, 191, 190, 319, 178, 179, 160, 161, 162, 163, 170, 171, 175, 176, 191, 190, 319, 320, 321, 330, 329, 150, 149, 148, 130, 136, 128, 126, 87, 86, 85, 57, 58, 59, 58, 59, 60, 61, 60, 62, 63, 30, 31, 40, 47, 57, 58, 59, 60, 62, 63, 64, 65, 64, 66, 67, 65, 67, 66, 68, 69, 68, 70, 71, 5, 6, 7, 8, 9, 10, 93, 102, 103, 104, 76, 79, 80, 62, 65, 79, 80, 81, 82, 83, 85, 57, 58, 59, 60, 81, 88, 87, 89, 91, 92, 94, 10, 93, 90, 88, 87, 89, 91, 92, 90, 109, 107, 103, 8, 9, 10, 11, 12, 97, 98, 99, 100, 101, 89, 91, 92, 94, 95, 98, 99, 100, 101, 91, 92, 94, 95, 98, 99, 100, 110, 87, 89, 91, 92, 94, 95, 98, 113, 13, 116, 117, 114, 111, 112, 113, 13, 116, 117, 114, 115, 116, 117, 118, 119, 121, 193, 192, 193, 192, 198, 182, 199, 183, 184, 216, 215, 216, 223, 187, 253, 254, 188, 241, 239, 238, 237, 236, 185, 258, 257, 244, 236, 234, 233, 221, 240, 221, 233, 224, 222, 219, 220, 221, 238, 242, 241, 246, 247, 248, 249, 215, 216, 223, 250, 249, 245, 239, 240, 217, 215, 217, 218, 201, 202, 203, 204, 209, 208, 205, 204, 209, 208, 207, 206, 205, 204, 209, 208, 207, 313, 312, 144, 138, 132, 125, 124, 135, 141, 315, 206, 205, 204, 209, 231, 230, 226, 225, 204, 209, 231, 230, 226, 225, 231, 230, 294, 295, 296, 211, 311, 145, 144, 138, 132, 131, 137, 138, 139, 140, 134, 133, 139, 143, 142, 140, 134, 133, 132, 131, 128, 126, 125, 124, 123, 122, 120, 121, 193, 194, 195, 196, 182, 199, 200, 201, 318, 122, 120, 118, 119, 192, 198, 197, 194, 195, 196, 182, 199, 183, 212, 213, 214, 184, 216, 223, 250, 251, 252, 187, 253, 269, 189, 188, 255, 256, 244, 243, 242, 241, 246, 247, 248, 251, 252, 187, 269, 189, 268, 267, 264, 256, 259, 262, 261, 258, 257, 244, 243, 242, 241, 246, 247, 252, 187, 269, 270, 272, 273, 271, 268, 267, 264, 266, 265, 263, 262, 261, 260, 291, 288, 289, 290, 292, 293, 292, 290, 296, 211, 210, 209, 231, 230, 226, 222, 219, 220, 221, 240, 217, 218, 201, 202, 203, 316, 135, 134, 133, 132, 125, 124, 123, 317, 202, 219, 220, 224, 227, 228, 229, 294, 293, 235, 185, 258, 259, 262, 285, 284, 263, 264, 266, 275, 276, 265, 283, 280, 281, 284, 283, 282, 277, 278, 279, 280, 308, 281, 286, 287, 261, 260, 293, 235, 228, 229, 294, 295, 292, 291, 288, 298, 299, 289, 290, 300, 290, 300, 301, 300, 301, 302, 303, 299, 300, 301, 302, 303, 304, 305, 297, 287, 288, 298, 304, 305, 297, 298, 304, 305, 306, 186, 147, 136, 128, 126, 87, 89, 91, 96, 11, 12, 97, 98, 99, 92, 94, 95, 96, 11, 12, 97, 96, 11, 12, 13, 116, 117, 114, 111, 110, 87, 86, 84, 83, 58, 57, 56, 57, 56, 157, 56, 157, 156, 126, 87, 86, 85, 57, 56, 157, 158, 157, 158, 54, 55, 49, 51, 53, 52, 50, 48, 45, 42, 37, 34, 33, 32, 31, 30, 27, 22, 17, 16, 15, 14, 2, 1]

Fonte: O autor (2020)

Figura 29 – Quantidade de nós para o grafo do Cordeiro



Fonte: O autor (2020).

O aumento do grau de um nó, como já discutido no capítulo anterior, aumenta a quantidade de mudanças de transição possíveis para quebra de *tours*. Logo, percebe-se pela Figura 29, que para o bairro Cordeiro haverá um esforço computacional maior por apresentar nós de grau 12, 14 e 18, embora que em menor quantidade comparado aos nós de grau 2 a 8.

Os dados de entrada incorporados ao modelo para os bairros Engenho do Meio e Cordeiro apresentaram a estimativa da demanda total de lixo que deve ser coletado, sendo 7.295,53 kg e 18.062,92 kg, respectivamente. Portanto, como a demanda de lixo é muito superior para apenas um veículo realizar a coleta geral, então se faz necessário a utilização da heurística *TourBreakHeuristic*. A resolução do problema pela heurística proposta apresentou os seguintes dados presentes nas Tabela 3 e Tabela 4, onde se pode verificar a solução, tempo computacional e alguns comportamentos dos dados para 10 testes seguidos para cada bairro e comparação com as distâncias de rotas dada pela empresa responsável pela coleta de lixo no município.

Segundo os dados da Tabela 3, a resolução se mostrou rápida para o bairro Engenho do Meio, com média do tempo de resolução em torno de 7,243 segundos e com três nós sendo selecionados dentre as 10 respostas apresentadas (58, 75 e 86), com três valores para a função objetivo (soma das distâncias totais). Para cada rota tem-se a distância do veículo sair da garagem, fazer a coleta nas ruas do bairro e depositar o lixo no aterro sanitário municipal.

Tabela 3 – Resultado do TourBreakHeuristic para o bairro Engenho do Meio

#	Valor da F. O. (km)	Quebra no nó	Distâncias totais (Km)		Demanda de lixo (Ton.)		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	23,683	86	11,879	11,804	3,692	3,603	7,5138
2	23,337	75	11,706	11,631	3,812	3,483	7,2638
3	23,337	75	11,706	11,631	3,812	3,483	7,0607
4	23,745	58	11,910	11,835	3,831	3,464	7,2325
5	23,337	75	11,706	11,631	3,812	3,483	7,2013
6	23,337	75	11,706	11,631	3,812	3,483	7,3731
7	23,683	86	11,879	11,804	3,692	3,603	7,5138
8	23,745	58	11,910	11,835	3,831	3,464	7,2325
9	23,337	75	11,706	11,631	3,812	3,483	6,9826
10	23,337	75	11,706	11,631	3,812	3,483	7,0607

Fonte: O autor (2020).

O menor valor obtido foi na quebra do nó 75, com um valor total igual a 23,337 km de percurso para os dois veículos atenderem todo o bairro. Tal valor se mostra mais interessante do que a estipulada pela empresa que realiza o serviço na cidade, visto que foi informado que cada veículo percorria aproximadamente 15 km por rota. Embora este dado seja apenas estimado pela empresa local, serve de comparação para verificar que a resposta da heurística proposta é bem melhor do que a empírica.

Para a resolução do bairro Cordeiro, presente na Tabela 4, a resolução não se mostrou muito rápida, com tempo médio de resolução de 6,18 minutos e com três pares de nós selecionados entre as 10 respostas geradas (40-98, 75-215 e 36-115). Cada solução gerou 3 rotas viáveis para o problema. Análoga a resolução do Engenho do Meio, para cada rota tem-se a distância total do veículo sair da sua garagem, realizar a coleta nas ruas selecionadas e depositar o lixo no aterro sanitário da cidade.

Tabela 4 – Resultado do TourBreakHeuristic para o bairro Cordeiro

#	Valor da F. O. (km)	Quebra nos nós	Distâncias totais (Km)			Demanda de lixo (Ton.)			Tempo de resolução (seg.)
			Rota 1	Rota 2	Rota 3	WT1	WT2	WT3	
1	92,20	40 – 98	30,677	30,796	30,727	6,095	6,210	5,757	330,069
2	78,70	75 – 215	26,177	26,296	26,227	5,984	6,132	5,946	384,950
3	92,20	40 – 98	30,677	30,796	30,727	6,095	6,210	5,757	315,476
4	82,87	36 – 115	27,567	27,686	27,617	5,789	6,198	6,075	411,668
5	82,87	36 – 115	27,567	27,686	27,617	5,789	6,198	6,075	400,102
6	78,70	75 – 215	26,177	26,296	26,227	5,984	6,132	5,946	436,530
7	92,20	40 – 98	30,677	30,796	30,727	6,095	6,210	5,757	330,069
8	92,20	40 – 98	30,677	30,796	30,727	6,095	6,210	5,757	356,530
9	78,70	75 – 215	26,177	26,296	26,227	5,984	6,132	5,946	411,668
10	82,87	36 – 115	27,567	27,686	27,617	5,789	6,198	6,075	330,069

Fonte: O autor (2020)

Devido a rota atual para o bairro Cordeiro não ultrapassar o percurso total de 30 km (segundo informação da empresa), então as quebras nos nós 40 e 98 são viáveis, porém não são atrativas para a empresa, por apresentarem rotas com percurso superior ao atual estimado. A função objetivo (soma das distâncias totais) teve menor resultado para a quebra nos nós 75 e 215 com distância total dos três veículos coletores igual a 78,70 km. Esta solução é uma boa opção para a rota atual, que apresentou rotas menores do que a empírica informada pela empresa.

A nível de comparação, os dados dos bairros foram resolvidos pelo modelo PLIM de Golden e Wong (1981), cuja soluções aparecem na Tabela 5 para o bairro Engenho do Meio e na Tabela 6 para o bairro Cordeiro. Embora o modelo PLIM seja exato, este foi resolvido 10 vezes para comparar a média de seu tempo computacional com o da heurística proposta. O valor da Função Objetivo (soma das distâncias das rotas geradas) foi comparado com a *TourBreakHeuristic* por meio do *gap*, calculado da seguinte forma: *gap* = ((soma das distâncias das rotas da heurística – solução PLIM) / solução PLIM).

Comparando os resultados para o bairro Engenho do Meio (Tabela 5), obtém-se que a heurística chegou ao ótimo em 6 das 10 rodadas feitas para a quebra no nó 75. O pior caso para a solução heurística foi para um *gap* de 1,75%. O tempo para resolução do modelo PLIM se mostrou bem superior ao da heurística, com média de tempo de 6,46 minutos, enquanto que a resolução do procedimento proposto resolveu em menos de 8 segundos. Para isso, se mostrou interessante resolver a *TourBreakHeuristic* mais de uma vez para comparar as soluções e garantir uma solução melhor.

Tabela 5 – TourBreakHeuristic x Modelo PLIM: bairro Engenho do Meio

Quebra no nó	*Tour Break Heuristic*		Modelo PLIM		*Gap* (%)
	F. Obj.	Tempo de resolução (seg.)	F. Obj.	Tempo de resolução (seg.)	
86	23.683	7,514			1,48%
75	23.337	7,157	23,337	387,310	0,00%
58	23.745	7,233			1,75%

Fonte: O autor (2020)

Comparando os resultados presentes na Tabela 6, se verifica que a heurística obteve sua melhor solução com *gap* de 6,06% comparado ao modelo exato para quebras nos nós 75 e 215 com uma diferença de distância de 4,5 km. O pior caso para a solução heurística foi para um *gap* de 24,26%. Uma justificativa para este último *gap* é que embora o valor da função objetivo seja alto, mas suas rotas são tão balanceadas quanto as outras selecionadas. O tempo para resolução do modelo PLIM se mostrou superior ao da heurística proposta com uma média de tempo de 11,18 minutos. Embora o tempo de resolução da heurística também não seja baixo, mas sua resolução é quase 55% o tempo que se gasta para resolver o modelo PLIM.

Tabela 6 – TourBreakHeuristic x Modelo PLIM: bairro Cordeiro

Quebra nos nós	*Tour Break Heuristic*		Modelo PLIM		*Gap* (%)
	F. Obj.	Tempo de resolução (seg.)	F. Obj.	Tempo de resolução (seg.)	
40 - 98	92,20	333,036			24,26%
75 - 215	78,70	411,049	74,200	670,937	6,06%
36 - 115	82,87	380,613			11,68%

Fonte: O autor (2020)

Por fim, pode-se verificar que o modelo heurístico proposto obteve bons resultados comparados ao modelo ótimo descrito por Golden e Wong (1981) e bem melhores do que os empíricos utilizados atualmente pela empresa da cidade. Uma consideração a ser feita é que embora o modelo PLIM garanta o ótimo, sua aplicação prática não se torna viável para os bairros, visto que algumas ruas são percorridas até 5 vezes na resolução do PCC direcionado, o que para o modelo de Golden e Wong (1981) obrigaria que 5 caminhões coletores deveriam ser necessários para realizar o percurso ótimo e no contexto prático, 5 caminhões em atividade

aumenta o custo para a empresa, referente a salários de mais motoristas, o aumento do combustível, o desgaste de peças e a maior probabilidade de quebra dos veículos. Assim, a heurística garante o mínimo possível de veículos coletores para atender a demanda de lixo coletado para cada rota.

6.2 INSTÂNCIAS CRIADAS PARA VERIFICAÇÃO DOS DADOS

Baseado na elaboração da heurística proposta, algumas instâncias foram definidas para analisar o comportamento da *Tour Break Heuristic* com diferentes grafos direcionados, sendo motivo de estudo e discussão para esse subcapítulo.

Os grafos foram chamados de: n31a51 ($|N| = 31$ e $|A| = 51$), n180a312 ($|N| = 180$ e $|A| = 312$) e n315a527 ($|N| = 315$ e $|A| = 527$). Tais grafos direcionados foram gerados a partir de uma estrutura com arcos apresentando ligações aleatórias, porém garantindo a relação de conectividade entre os nós.

Os resultados de tempo computacional para os passos 1 e 2 do procedimento proposto estão representados na Tabela 7. 10 resoluções seguidas foram realizadas para cada instância. O tempo computacional refere-se à criação do arquivo ".lp" e resolução do PCC direcionado pelo *SoPlex*. Estes dados mostram a rápida resposta para as instâncias em suas resoluções, estando todas abaixo de 1 segundo.

Tabela 7 – Tempo computacional para as instâncias geradas nas etapas 1 e 2

Instância	**Tempo Computacional (seg.)**										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Média**
n31a51	0,324	0,269	0,270	0,263	0,300	0,247	0,270	0,247	0,247	0,238	**0,266**
n180a312	0,464	0,386	0,369	0,369	0,423	0,468	0,385	0,448	0,385	0,369	**0,385**
n315a527	0,555	0,470	0,523	0,485	0,485	0,539	0,485	0,501	0,501	0,516	**0,501**

Fonte: O autor (2020)

A etapa 3 serviu apenas para ser um dado de entrada para a etapa 4, então não será aprofundada aqui, sendo que o *tour* Euleriano encontrado se mostra trivial para grafos balanceados. Então, dá-se ênfase aos resultados da etapa 4 com a resolução do *Tour Break Heuristic* e comparação com o modelo exato implementado, por meio do cálculo do *gap* = ((soma das distâncias das rotas da heurística – solução PLIM) / solução PLIM). Essa análise foi realizada para cada instância alterando a capacidade máxima de cada *tour* (W_T), de modo a analisar as soluções encontradas e seu esforço computacional.

Para a instância n31a51, a solução ao fim da etapa 2 deixou o grafo com 76 arcos para torná-lo balanceado. A demanda total a ser atendida no grafo é de 269. As tabelas a seguir mostram os resultados da heurística para os 10 resultados, variando W_T e anotando os nós que geram as quebras de *tours*, assim como o tempo computacional e o valor da Função Objetivo (soma das distâncias das rotas geradas). Assim, a Tabela 8 para W_T = 120, Tabela 9 para W_T = 150 e Tabela 10 para W_T = 268.

Tabela 8 - Resolução para o grafo n31a51 com W_T = 120

#	Valor da F. O.	Quebra nos nós	Distâncias			Demanda de lixo			Tempo de resolução (seg.)
			Rota 1	Rota 2	Rota 3	WT1	WT2	WT3	
1	**388**	7 - 14	130	130	128	116	81	72	0,385
2	391	14 - 2	131	131	129	116	86	67	0,323
3	**388**	7 - 14	130	130	128	116	81	72	0,322
4	**388**	7 - 14	130	130	128	116	81	72	0,423
5	393	14 - 7	130	132	131	81	116	72	0,501
6	391	2 - 14	129	131	131	67	86	116	0,385
7	391	2 - 14	129	131	131	67	86	116	0,332
8	**388**	7 - 14	130	130	128	116	81	72	0,401
9	391	2 - 14	129	131	131	67	86	116	0,423
10	**388**	7 - 14	130	130	128	116	81	72	0,385

Fonte: O autor (2020)

A solução com a capacidade W_T = 120 apresentou 2 mudanças de transição que designou 3 rotas viáveis. Não houve solução única, com quebras nos nós 7 e 14, 14 e 2, 14 e 7 e, 14 e 2. Percebe-se que quebrar uma rota em 7 e depois em 14, pode não ser o mesmo do que quebrar a rota em 14 e depois em 7, devido a segunda mudança de transição estar atrelado a uma rota diferente, isso pode ser percebido nos pesos das demandas para as rotas 1 e 2. A melhor solução encontrada está destacada em negrito e foi obtida com a quebra em 7 e 14. O tempo computacional se mostrou muito baixo, inferior a 1 segundo.

Para W_T = 150 são necessárias apenas 2 rotas (Tabela 9). Duas soluções foram apresentadas com uma mudança de transição em cada. A melhor solução (em negrito) foi uma quebra de *tour* no nó 14, que apareceu como melhor solução em 6 das 10 resoluções. Percebe-se que 13 é uma solução viável por apresentar uma diferença média absoluta igual a quebra em 14, porém a soma de suas rotas é mais extensa.

Tabela 9 - Resolução para o grafo n31a51 com W_T = 150

#	Valor da F. O.	Quebra no nó	Distâncias		Demanda de lixo		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	**388**	14	242	146	140	129	0,332
2	400	13	248	152	135	134	0,369
3	**388**	14	242	146	140	129	0,401
4	**388**	14	242	146	140	129	0,363
5	400	13	248	152	135	134	0,385
6	**388**	14	242	146	140	129	0,385
7	400	13	248	152	135	134	0,363
8	**388**	14	242	146	140	129	0,354
9	**388**	14	242	146	140	129	0,384
10	400	13	248	152	135	134	0,410

Fonte: O autor (2020)

Tabela 10 - Resolução para o grafo n31a51 com W_T = 268

#	Valor da F. O.	Quebra no nó	Distâncias		Demanda de lixo		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	388	9	160	228	99	170	0,085
2	388	9	160	228	99	170	0,100
3	388	9	160	228	99	170	0,147
4	388	9	160	228	99	170	0,138
5	388	9	160	228	99	170	0,137
6	388	9	160	228	99	170	0,137
7	388	9	160	228	99	170	0,131
8	388	9	160	228	99	170	0,085
9	388	9	160	228	99	170	0,085
10	388	9	160	228	99	170	0,163

Fonte: O autor (2020)

Para capacidade superior, como 268, a solução apresentou solução única, com quebra no nó 9 e um tempo computacional inferior às outras capacidades. Isto se justifica por que aumentar a capacidade do veículo implica em aumentar o espaço de busca para soluções viáveis na Etapa 4 do algoritmo, que irá gerar mais soluções viáveis, porém com uma única solução apresentando uma diferença média absoluta menor. A quebra no nó 9 não foi solução para o caso de W_T = 150, devido a segunda rota apresentar capacidade igual a 170.

Agora, serão comparadas as soluções heurísticas com o modelo PLIM implementado, que também foi resolvido 10 vezes, apenas para comparação do tempo de resolução. A Tabela 11 exibe essa comparação com W_T igual a 120, 150 e 268, respectivamente. Para cada

capacidade máxima foi utilizada apenas as soluções diferentes e o tempo computacional apresenta a média da quantidade de vezes que esta foi resolvida.

Tabela 11 – Instância n31a51: Comparação de resultados com modelo PLIM

W_T	*Tour Break Heuristic*			Modelo PLIM		*Gap* (%)
	Quebra no(s) nó(s)	F. Obj.	Tempo de resolução (seg.)	F. Obj.	Tempo de resolução (seg.)	
120	7 - 14	388	0,385	388	0,145	0,00%
	14 - 2	391	0,323			0,77%
	14-7	393	0,501			1,29%
	2-14	391	0,385			0,77%
150	14	388	0,370	388	0,227	0,00%
	13	400	0,382			3,09%
268	9	388	0,121	388	0,227	0,00%

Fonte: O autor (2020).

Os dados da Tabela 11 mostram que: para W_T igual a 120, o ótimo foi obtido em 5/10 rodadas e o pior caso para o *gap* de 1,29%; para W_T igual a 150, alcançou-se o ótimo em 6/10 resoluções, sendo o pior caso com um *gap* de 3,09%; já para a capacidade máxima de 268, o *gap* foi 0 para todas as rodadas. Somente para W_T igual a 268 o tempo de resolução da heurística foi melhor do que a resolução do modelo exato.

A instância n180a312 possui uma demanda total de 3.919 e o grafo balanceado oriundo do final do passo 2 possui 626 arcos. Os resultados da heurística para 10 resoluções são apresentados para W_T igual a 1.500 (Tabela 12), 2.000 e 3.000 (Tabela 13) e 3.918 (Tabela 14).

Tabela 12 - Resolução para o grafo n180a312 com W_T = 1.500

#	Valor da F. O.	Quebra nos nós	Distâncias			Demanda de lixo			Tempo de resolução (seg.)
			Rota 1	Rota 2	Rota 3	WT1	WT2	WT3	
1	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	138,757
2	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	157,664
3	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	139,170
4	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	131,003
5	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	143,413
6	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	143,159
7	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	128,643
8	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	161,438
9	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	129,578
10	7.451	20 - 140	2.500	2.470	2.481	1.089	1.441	1.389	128,251

Fonte: O autor (2020)

A Tabela 12 apresentou os resultados para a instância n180a312 com W_T = 1.500, que para o cálculo de V apresentou valor igual a 3 veículos/rotas. Para as 10 resoluções, houve apenas uma resposta com quebra nos nós 20 e 140. Este resultado teve um esforço computacional maior com tempo médio de resolução de 140,1 segundos.

Tabela 13 - Resolução para o grafo n180a312 com W_T variando entre 2.000 e 3.000

#	Valor da F. O.	Quebra no nó	Distâncias		Demanda de lixo		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	7.611	120	3.802	3.809	1.701	2.218	161,589
2	7.485	119	3.739	3.746	1.695	2.224	161,437
3	7.611	121	3.809	3.802	2.218	1.701	145,178
4	7.611	120	3.802	3.809	1.701	2.218	147,063
5	7.611	121	3.809	3.802	2.218	1.701	149,152
6	7.485	119	3.739	3.746	1.695	2.224	155,310
7	**7.451**	74	3.722	3.729	1.695	2.224	121,273
8	**7.451**	74	3.722	3.729	1.695	2.224	133,655
9	**7.451**	74	3.722	3.729	1.695	2.224	147,225
10	7.611	121	3.809	3.802	2.218	1.701	163,974

Fonte: O autor (2020)

A resolução presente na Tabela 13 apontam os resultados para W_T igual a 2.000 e 3.000 na mesma tabela por apresentarem resoluções iguais para ambos, assim, os resultados embora mostrem valores para uma capacidade máxima específica, mas qualquer valor dentro deste intervalo (2.000 e 3.000) geram as mesmas soluções e um esforço computacional bem semelhante, além de serem suficientes apenas 2 veículos/rotas. As soluções geraram 4 mudanças de transição diferente para formar duas rotas viáveis (74, 119, 120 e 121), porém com um limite inferior presente apenas na quebra do nó 74, destacado em negrito.

A próxima análise é para W_T = 3.918. O principal motivo para introduzir esta capacidade bem específica é devido o interesse de verificar o quão grande uma capacidade deve se tornar para reduzir o espaço de busca e gerar soluções viáveis mais rápido, desde que esta capacidade não seja igual ou maior do que a demanda total de uma única rota, visto que isso inviabiliza a quebra de rotas. Então, utilizou-se a maior capacidade para quebrar um *tour* em dois, de modo a verificar sua resolução, como está presente nos resultados da Tabela 14, que encontrou uma única solução com quebra no nó 74 e um tempo computacional muito inferior a outros resultados da instância com capacidades diferentes.

Tabela 14 – Resolução para o grafo n180a312 com W_T = 3.918

#	Valor da F. O.	Quebra no nó	Distâncias		Demanda de lixo		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	7.451	74	3.722	3.729	1.695	2.224	1,203
2	7.451	74	3.722	3.729	1.695	2.224	1,224
3	7.451	74	3.722	3.729	1.695	2.224	1,156
4	7.451	74	3.722	3.729	1.695	2.224	1,172
5	7.451	74	3.722	3.729	1.695	2.224	1,150
6	7.451	74	3.722	3.729	1.695	2.224	1,156
7	7.451	74	3.722	3.729	1.695	2.224	1,172
8	7.451	74	3.722	3.729	1.695	2.224	1,157
9	7.451	74	3.722	3.729	1.695	2.224	1,156
10	7.451	74	3.722	3.729	1.695	2.224	1,234

Fonte: O autor (2020)

Os resultados gerados foram comparados com o modelo PLIM implementado, que por sua vez, foi resolvido para 9 veículos, devido alguns arcos terem sido multiplicados por 9 para tornar o grafo balanceado no Passo 2 do procedimento. Mesmo com esta resolução, obteve-se apenas a comparação com o resultado da função objetivo e comparado com o tempo computacional, presentes na Tabela 15.

Tabela 15 – Instância n180a312: Comparação de resultados com modelo PLIM

W_T	*Tour Break Heuristic*			Modelo PLIM		*Gap* (%)
	Quebra no(s) nó(s)	F. Obj.	Tempo de resolução (seg.)	F. Obj.	Tempo de resolução (seg.)	
1.500	7 - 14	7.451	140,107	7.451	337,057	0,00%
2.000	74	7.451	134,050	7.451	387,310	0,00%
	121	7.611	158,370			2,15%
3.000	120	7.611	154,321			2,15%
	119	7.485	152,760			0,46%
3.918	74	7.451	1,178	7.451	312,803	0,00%

Fonte: O autor (2020)

Com as comparações, pôde-se verificar que para W_T igual a 1.500 e 3.918 todas as resoluções obtiveram o ótimo em suas respectivas quebras. Para a capacidade máxima em 2.000 e 3.000 o ótimo foi atrelado a quebra no nó 74 também presente em W_T igual a 3.918, porém com várias soluções, sendo o pior caso com um *gap* de 2,15%, o que representa uma diferença

de 160 unidades. Os tempos computacionais da heurística para a capacidade máxima de 1.500, 2.000 e 3.000 foram considerados altos, ainda que menores do que os do resultado do modelo exato (entre 59% e 65% menor).

De forma análoga, os dados obtidos com a heurística para a instância n315a527 após o passo 2 obteve 779 arcos e demanda total de 3.578. A instância foi resolvida para capacidades máximas de 1.500, 2.000 e 3.577 nas respectivas, Tabela 16, Tabela 17 e Tabela 18.

Tabela 16 - Resolução para o grafo n315a527 com W_T = 1.500

#	Valor da F. O.	Quebra nos nós	Distâncias			Demanda de lixo			Tempo de resolução (seg.)
			Rota 1	Rota 2	Rota 3	WT1	WT2	WT3	
1	**8.603**	254 - 68	2.868	2.856	2.879	1.109	1.226	1.243	231,295
2	**8.603**	254 - 68	2.868	2.856	2.879	1.109	1.226	1.243	179,646
3	8.833	276 - 68	2.958	2.940	2.935	1.246	1.219	1.113	182,948
4	8.929	68 - 282	2.982	2.962	2.985	1.243	1.224	1.111	185,470
5	8.833	276 - 68	2.958	2.940	2.935	1.246	1.219	1.113	179,263
6	**8.603**	254 - 68	2.868	2.856	2.879	1.109	1.226	1.243	180,607
7	**8.603**	68 - 254	2.879	2.856	2.868	1.243	1.226	1.109	179,322
8	**8.603**	254 - 68	2.868	2.856	2.879	1.109	1.226	1.243	159,378
9	8.833	276 - 68	2.958	2.940	2.935	1.246	1.219	1.113	170,700
10	**8.603**	254 - 68	2.868	2.856	2.879	1.109	1.226	1.243	191,989

Fonte: O autor (2020)

Para uma capacidade máxima de 1.500, o valor de V é igual a 3 veículos/rotas. A Tabela 16 apresentou 4 quebras diferentes com 3 rotas viáveis cada. Observa-se que, neste caso, a quebra primeiro em 254 e depois em 68 teve o mesmo valor do que quebrar o *tour* em 68 primeiro e depois em 254, o qual ocasionou no menor valor para a função objetivo. O tempo computacional foi alto, embora a instância seja grande compara da as demais.

Para W_T = 2.000 o parâmetro V é igual a 2 veículos/rotas. Os resultados da Tabela 17 mostram três soluções diferentes com quebras em 110, 145 e 146 determinando duas rotas viáveis. A mudança de transição no nó 110 obteve o menor valor da função objetivo em 4 das 10 rodadas. As demais soluções são soluções viáveis, devido apresentarem o mesmo valor da diferença absoluta média entre as distâncias das rotas.

Tabela 17 - Resolução para o grafo n315a527 com W_T = 2.000

#	Valor da F. O.	Quebra no nó	Distâncias		Demanda de lixo		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	9.001	146	4.504	4.497	1.984	1.594	190,361
2	8.747	145	4.377	4.370	1.989	1.589	227,579
3	**8.603**	110	4.305	4.298	1.998	1.580	213,204
4	9.001	146	4.504	4.497	1.984	1.594	230,776
5	**8.603**	110	4.305	4.298	1.998	1.580	227,487
6	8.747	145	4.377	4.370	1.989	1.589	218,141
7	9.001	146	4.504	4.497	1.984	1.594	222,330
8	**8.603**	110	4.305	4.298	1.998	1.580	226,821
9	**8.603**	110	4.305	4.298	1.998	1.580	227,200
10	8.747	145	4.377	4.370	1.989	1.589	229,638

Fonte: O autor (2020)

Para a capacidade máxima igual a 3.577 (Tabela 18), tem-se um limite máximo para garantir a necessidade de quebra de *tour* (V = 2 veículos/rotas). Com isso, a Tabela 18 mostra um tempo computacional baixo para encontrar soluções viáveis, onde foi encontrado três quebras diferentes (108, 109 e 146), com a quebra em 109 gerando uma solução melhor do que as demais, embora todas apresentem o mesmo valor da diferença absoluta média.

Tabela 18 - Resolução para o grafo n315a527 com W_T = 3.577

#	Valor da F. O.	Quebra no nó	Distâncias		Demanda de lixo		Tempo de resolução (seg.)
			Rota 1	Rota 2	WT1	WT2	
1	8.603	109	4.305	4.298	2.007	1.571	1,468
2	9.001	146	4.504	4.497	1.984	1.594	1,434
3	8.603	109	4.305	4.298	2.007	1.571	1,526
4	8.603	109	4.305	4.298	2.007	1.571	1,265
5	8.603	109	4.305	4.298	2.007	1.571	1,103
6	8.613	108	4.310	4.303	2.005	1.573	1,203
7	9.001	146	4.504	4.497	1.984	1.594	1,234
8	9.001	146	4.504	4.497	1.984	1.594	1,156
9	8.603	109	4.305	4.298	2.007	1.571	1,118
10	9.001	146	4.504	4.497	1.984	1.594	1,109

Fonte: O autor (2020)

A Tabela 19 apresenta a comparação dos resultados (valor da função objetivo e tempo de resolução) para a instância n315a527 com o resultado do modelo PLIM.

Tabela 19 – Instância n315a527: Comparação de resultados com modelo PLIM

W_T	*Tour Break Heuristic*			Modelo PLIM		*Gap* (%)
	Quebra no(s) nó(s)	**F. Obj.**	**Tempo de resolução (seg.)**	**F. Obj.**	**Tempo de resolução (seg.)**	
1.500	254 - 68	8.603	179,646	8.603	73,165	0,00%
	276 - 68	8.833	182,948			2,67%
	68 - 282	8.929	185,470			3,79%
	68 - 254	8.603	179,322			0,00%
2.000	110	8.603	223,678	8.603	785,011	0,00%
	145	8.747	225,119			1,67%
	146	9.001	214,489			4,63%
3.577	108	8.613	1,203	8.603	832,247	0,12%
	109	8.603	1,296			0,00%
	146	9.001	1,233			4,63%

Fonte: O autor (2020)

Analisando os resultados presentes na Tabela 19, pôde-se verificar que para W_T = 1.500, 2 das 4 quebras diferentes apresentaram o ótimo (254-68 e 68-254), com um pior *gap* de 3,79% e um tempo de resolução muito pior do que o executado pelo modelo exato. Para W_T = 2.000, a melhor quebra para duas rotas viáveis ocorreu no nó 110, o pior *gap* para esta capacidade foi de 4,63%, igual em W_T = 3.577, e o tempo de execução foi bem inferior comparado ao modelo PLIM, embora ainda alto. Por fim, para W_T = 3.577, o ótimo foi encontrado em 1 das 3 soluções possíveis pela *TourBreakHeuristic* e todas as resoluções apresentaram um tempo computacional muito baixo, sendo cerca de 15% da resolução para o modelo exato.

Diante dos pontos analisados, considera-se que a heurística teve boas chances de alcançar o ótimo com algumas resoluções a mais e com um tempo computacional de resolução melhor, na maioria dos casos, do que o modelo exato proposto por Golden e Wong (1981). Entretanto, mais testes precisariam ser feitos para avaliar ainda o quanto uma solução boa pode custar em termos de tempo de resolução e qual a influência do aumento da capacidade máxima do veículo coletor para o tempo de execução do procedimento.

7 CONSIDERAÇÕES FINAIS

Entre as variações do Problema do Carteiro Chinês (PCC) encontradas na literatura, o Problema do Carteiro Chinês Capacitado (PCCC) por incluir demandas (pesos) que devem ser servidas nos arcos uma única vez, apresenta um contexto prático, que é muito abordado tanto na literatura quanto nas aplicações reais. Entre essas aplicações está a coleta de lixo urbano, que com o passar dos anos e o aumento da população tem se tornado um dos enormes problemas das grandes cidades para os gestores públicos.

A motivação inicial deste trabalho foi a utilização do Modelo de Programação Inteira descrito por Wang e Wen (2002), que embora seja um modelo com variáveis inteiras, pode ser modelado com Programação Linear para resolução do PCC devido sua relação de total unimodularidade. Esse problema chamou a atenção, porém devido sua fácil aplicação, foi proposto inserir demandas aos arcos e verificar rotas viáveis para mais de um veículo. Outra motivação foi a utilização do *software OpenStreetMap*, que consegue modelar qualquer conjunto de dados geográficos em um grafo a partir de um cadastro de pontos.

O PCCC por apresentar complexidade difícil, abre espaço para propostas heurísticas, para sua resolução, que embora não garanta o ótimo, consegue determinar soluções melhores do que as empíricas e com um tempo de resolução bem inferior. Devido esta abordagem, este trabalho procurou apresentar um procedimento heurístico para resolução do PCCC de modo que se determine rotas viáveis para veículos coletores de lixo urbanos respeitando suas capacidades.

O procedimento foi dividido em 4 passos que foram bem descritos e cuja etapa 4 corresponde a um procedimento que determina mais de uma rota por meio de mudanças de transição nos nós, garantindo a quebra de um único *tour* em dois ou mais. O algoritmo heurístico foi implementado e aplicado para dois bairros da cidade de Recife-PE (Engenho do Meio e Cordeiro) e para três instâncias artificiais, a fim de se comparar seus resultados aos do modelo de Programação Linear Inteira Mista (PLIM) descrito na literatura por Golden e Wong (1981).

Para os bairros, a heurística se mostrou bem aplicada e com bons resultados chegando a solução ótima para 6 das 10 resoluções feitas para o bairro Engenho do Meio com um tempo de resolução inferior a 8 segundos para a quebra no nó 75. Já para o bairro Cordeiro, a melhor solução encontrada pela heurística atingiu um *gap* de 6,06% comparado ao modelo PLIM, porém com um tempo de resolução próximo dos 55% do que é gasto para resolver o modelo ótimo.

As soluções encontradas pela heurística para os bairros têm mais aplicabilidade prática do que o modelo PLIM de Golden e Wong (1981). Essa afirmação é devida ao modelo restringir que para cada arco que é percorrido mais de uma vez no PCC, deverá ser atendido por um veículo diferente, ou seja, se um segmento de rua precisar ser percorrido 5 vezes para garantir um *tour* Euleriano, então serão necessários 5 veículos para cada um percorrer o segmento de rua uma única vez. Isto na prática se torna inviável, porque quanto mais caminhões coletores estiverem em operação, maior será o custo com salário de funcionários, manutenção dos veículos, etc. Então, a não ser que um grafo tenha todos os seus nós com grau menor ou igual ao número de veículos desejado para as coletas, este modelo não tem viabilidade prática.

Entre os principais resultados encontrados, cita-se a contribuição científica para as heurísticas construtivas, em especial, o método *Route-First Cluster-Second*, com a nova abordagem para resolução do PCCC com foco na coleta de lixo; a contribuição para a sociedade com um programa que pode ser aplicável em qualquer contexto de coleta de lixo urbano e por usar um *software* com dados geográficos que podem ser manipulados (*OpenStreetMap*) tem aplicabilidade no mundo inteiro e ao econômico pela proposta de ter um programa que melhora as rotas para coletas de lixo e reduza os gastos nos cofres públicos, visto que se gasta muito com limpeza urbana no Brasil. E por fim, o programa que foi desenvolvido em linguagem de programação *Python* 3.7 é de fácil utilização e poderá se pensar em comercializá-lo para que empresas utilizem o serviço para melhorarem suas rotas de coleta de lixo urbano.

A heurística pode ser incorporada aos problemas de coleta de lixo urbano, porém, caso seja pertinente para outros tipos de aplicações que utilizem veículos com restrição de capacidade, o modelo também pode ser aplicado, aumentando a gama de aplicações do método. Este estudo se faz necessário para se avaliar se a heurística *Tour Break Heuristic* tem aplicabilidade em outros Problemas de Roteamento de Arcos.

O trabalho apresentou algumas limitações, sendo destaque para duas delas. Primeiro, por não apresentar dados precisos e quantitativos quanto as rotas de coleta de lixo urbano por meio da coleta de dados da empresa que executa a operação na cidade de Recife-PE, então não se pode comparar com exatidão o quão eficiente a heurística proposta é melhor do que a realizada atualmente. Mesmo assim, o parâmetro dado para o percurso máximo pertencente a cada rota (15 km para o Engenho do Meio e 30 km para o Cordeiro) já se mostrou um ganho comparado ao atual.

Segunda limitação da pesquisa refere-se ao cadastro dos nós (esquinas de segmentos de ruas) no *OpenStreetMap* que é um processo lento e cansativo, que pode estar sujeito a erro humano se este não estiver atento as ruas e aos pontos onde a coleta de lixo realmente ocorre

na prática. Para procurar reduzir este problema, é interessante desenvolver um programa na linguagem do programa *JAVA* que gere de forma mais rápida o grafo resultante ao bairro desejado.

Diante do exposto no trabalho, ficam algumas sugestões para trabalhos futuros com o intuito de melhorar a heurística, e assim, torná-la mais competitiva perante as já existentes. Para melhorar a solução da heurística, pode-se optar por realizar buscas locais da solução inicial para verificar se a solução melhora com mudanças de transição locais nos nós. Para procurar melhorar a resposta computacional pode-se optar por pensar em realizar mudanças de transição múltiplas para gerar mais de duas rotas viáveis (quando necessário), afim de acelerar a busca por soluções melhores ao invés de inicialmente quebrar em duas rotas e só depois em três rotas e assim por diante.

A nível de comparação, pensa-se em buscar comparar os resultados e tempo computacional da heurística proposta com outras abordagens para o método *Route-First Cluster-Second*, em especial os trabalhos de Vecchi et al. (2016), Vidal (2017) e Van Bevern, Komusiewicz e Sorge (2017).

Por fim, necessita-se analisar mais instâncias. Desde incluir grafos maiores (600 a 1.000 nós e 800 a 2.000 arcos) e continuar variando a capacidade máxima dos veículos, a fim de verificar o tempo de resposta e avaliar a relação do aumento da capacidade máxima com o tempo computacional do algoritmo.

REFERÊNCIAS

ABRELPE – Associação Brasileira de Empresas de Limpeza Pública e Resíduos Especiais. *Panorama dos resíduos sólidos no Brasil 2018/2019*. Disponível em: <http://abrelpe.org.br/download-panorama-2018-2019/>. Acesso em: 03 dez. 2019.

ACHTERBERG, T. SCIP: solving constraint integer programs. **Mathematical Programming Computation**, v. 1, n. 1, p. 1-41, 2009.

AHUJA, R.; MAGNANTI, T.; ORLIN, J. **Network Flow**: theory, algorithms, and applications. Upper Saddle River, New Jersey: Prentice Hall, 1993.

ALSTRUP, S. et al. A Hamiltonian Cycle in the Square of a 2-connected Graph in Linear Time. In: **Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms**. Society for Industrial and Applied Mathematics, p. 1645-1649, 2018.

AMBERG, A.; DOMSCHKE, W.; VOß, S. Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. **European Journal of Operational Research**, v. 124, n. 2, p. 360-376, 2000.

ARAKAKI, R. K.; USBERTI, F. L. Hybrid genetic algorithm for the open capacitated arc routing problem. **Computers & Operations Research**, v. 90, p. 221-231, 2018.

ARAKAKI, R. K.; USBERTI, F. L. An efficiency-based path-scanning heuristic for the capacitated arc routing problem. **Computers & Operations Research**, v. 103, p. 288-295, 2019.

ASSAD, A. A.; GOLDEN, B. L. Arc routing methods and applications. **Handbooks in operations research and management science**, v. 8, p. 375-483, 1995.

BABAEE TIRKOLAEE, E.; ALINAGHIAN, M.; BAKHSHI SASI, M.; SEYYED ESFAHANI, M. M. Solving a robust capacitated arc routing problem using a hybrid simulated annealing algorithm: a waste collection application. **Journal of Industrial Engineering and Management Studies**, v. 3, n. 1, p. 61-76, 2016.

BATISTA, G. V.; SCARPIN, C. T. Problema de roteamento em arcos capacitado e periódicos aplicados a um contexto real. **Produção Online**, v. 15, n. 3, p. 1080-1098, 2015.

BAUTISTA, J.; FERNÁNDEZ, E.; PEREIRA, J. Solving an urban waste collection problem using ants heuristics. **Computers & Operations Research**, v. 35, n. 9, p. 3020-3033, 2008.

BELENGUER, J. M.; BENAVENT, E. A cutting plane algorithm for the capacitated arc routing problem. **Computers & Operations Research**, v. 30, n. 5, p. 705-728, 2003.

BELIËN, J.; DE BOECK, L.; VAN ACKERE, J. Municipal solid waste collection and management problems: a literature review. **Transportation Science**, v. 48, n. 1, p. 78-102, 2012.

BELTRAMI, E.; BODIN, L. Networks and vehicle routing for municipal waste collection. **Networks**, v. 4, p. 65-94, 1974.

BERTSEKAS, D. P. **Network Optimization**: Continuous and Discrete Models. Belmont: Athena Scientific, 1998.

BODIN, L.; GOLDEN, B. Classification in vehicle routing and scheduling. **Networks**, v. 11, n. 2, p. 97-108, 1981.

BRANDÃO, J.; EGLESE, R. A deterministic tabu search algorithm for the capacitated arc routing problem. **Computers & Operations Research**, v. 35, n. 4, p. 1112-1126, 2008.

CAMPBELL, J. F.; CORBERÁN, A.; PLANA, I.; SANCHIS, J. M. Drone arc routing problems. **Networks**, v. 72, n. 4, p. 543-559, 2018.

CHEN, L.; HÀ, M. H.; LANGEVIN, A.; GENDREAU, M. Optimizing road network daily maintenance operations with stochastic service and travel times. **Transp. Res. Part E**, v. 64, p. 88-102, 2014.

CHEN, Y.; HAO, J. K.; GLOVER, F. A hybrid metaheuristic approach for the capacitated arc routing problem. **European Journal of Operational Research**, v. 253, n. 1, p. 25-39, 2016.

CHRISTOFIDES, N. The optimum traversal of a graph. **Omega**, v. 1, n. 6, p. 719-732, 1973.

CHRISTOFIDES, N**. Graph Theory**: An Algorithmic Approach. Academic Press, London, 1975.

CHRISTOFIDES, N.; BENAVENT, E.; CAMPOS, A.; CORBRAN, A.; MOTA, E. An optimal method for the mixed postman problem. **Proceedings of the 11th IFIP Conference**, Copenhagen, Denmark, p. 641-649, 1983.

CORBERÁN, A.; MARTÍ, R.; SANCHIS, J. M. A GRASP heuristic for the mixed Chinese postman problem. **European Journal of Operational Research**, v. 142, n. 1, p. 70-80, 2002.

CORBERÁN, A.; PRINS, C. Recent results on arc routing problems: An annotated bibliography. **Networks**, v. 56, n. 1, p. 50-69, 2010.

DOS SANTOS, C. G. *Uma proposta de modelagem matemática para um problema de roteirização periódica em arcos capacitados com múltiplas tarefas*. Tese (Doutorado). Programa de Pós-Graduação em Métodos Numéricos em Engenharia. Universidade Federal do Paraná. Curitiba, 2016.

DROR, M (Eds). **Arc routing**: theory, solutions and applications. Springer Science & Business Media, New York, 2012.

EDMONDS, J. The Chinese postman problem. **Operations Research**, 13, Supplement 1, B-73. 1965.

EDMONDS, J.; JOHNSON, E. L. Matching, Euler tours and the Chinese postman. **Mathematical programming**, v. 5, n. 1, p. 88-124, 1973.

EISELT, H.A.; GENDREAU, M.; LAPORTE, G. Arc routing problems. Part I: The Chinese postman problem. **Operations Research**, v. 43, n. 2, p. 231-242, 1995.

EULER, L.; Solutio Problematis ad Geometrian Situs Pertinentis. **Commentarii academiae scientarum Petropolitanae**, v. 8, p. 128-140, 1736.

EVANS, J. R.; MINIEKA, E. *Optimization algorithms for networks and graphs*. Marcel Dekker, 2 ed., New York, 1992.

FENG, L.; ONG, Y. S.; NGUYEN, Q. H.; TAN, A. H. Towards Probabilistic Memetic Algorithm: An Initial Study on Capacitated Arc Routing Problem. In: **IEEE Congress on Evolutionary Computation**. p. 1-7, 2010.

FORD, L. R.; FULKERSON, D. R. *Flows in network*. Princeton University Press, Princeton, New Jersey, 1962.

FOULDS, L.; LONGO, H.; MARTINS, J. A compact transformation of arc routing problems into node routing problems. **Annals of Operations Research**, v. 226, n. 1, p. 177-200, 2015.

FREDERICKSON, G. N. Approximation algorithms for some postman problems. **Journal of A.C.M.**, v. 26, n. 3, p. 538-554, 1979.

GANGA, G. M. D. *Trabalho de conclusão de curso (TCC) na engenharia de produção*: um guia prático de conteúdo e forma. São Paulo: Atlas, 2012.

GÁSPÁR, L.; BENCZE, Z. Salting route optimization in Hungary. **Transportation Research Procedia**, v. 14, p. 2421-2430, 2016.

GERHARDT, T. E.; SILVEIRA, D. T. *Métodos de pesquisa***.** Universidade Aberta do Brasil – UAB/UFRGS e Planejamento e Gestão para o Desenvolvimento Rural da SEAD/UFRGS. Porto Alegre: Editora da UFRGS, 2009.

GHIANI, G.; GUERRIERO, F.; IMPROTA, G.; MUSMANNO, R. Waste collection in Southern Italy: solution of a real-life arc routing problem. **International Transactions in Operational Research**, v. 12, n. 2, p. 135-144, 2005.

GHORPADE, T.; RANGARAJ, N. Order first split second heuristic for alternative routing strategy for freight railways. **Transport Policy**, 2019.

GIL, A. C. *Como elaborar projetos de pesquisa*. 5 ed. São Paulo: Atlas, 2010.

GLEIXNER, A. et al. The SCIP Optimization Suite 6.0. *ZIB-Report*, Zuse Institute Berlin, Berlin, Germany. p. 18-26, 2018. Disponível em: <http://www.optimization-online.org/DB_HTML/2018/07/6692.html>. Acesso em: 10 ago. 2019.

GODINHO FILHO, M.; JUNQUEIRA, R. A. R. Problema do Carteiro Chinês: escolha de métodos de solução e análise de tempos computacionais. **Produção**, v. 16, n. 3, p. 538-551, 2006.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear**: modelos e algoritmos. 2. ed. Elsevier, 2005.

GOLDEN, B. L.; WONG, R. T. Capacitated Arc Routing Problem. **Networks**, v. 11, p. 305–315, 1981.

GOUVEIA, L.; MOURÃO, M. C.; PINTO, L. S. Lower bounds for the mixed capacitated arc routing problem. **Computers & Operations Research**, v. 37, n. 4, p. 692-699, 2010.

GRAKOVA, E. et al. Waste Collection Vehicle Routing Problem on HPC Infrastructure. In: **IFIP International Conference on Computer Information Systems and Industrial Management**. Springer, Cham, p. 266-278, 2018.

GUAN, M. Graphic programming using odd or even points. **Chinese Math**, v. l, p. 273-277, 1962.

HAN, H.; PONCE CUETO, E. Waste Collection Vehicle Routing Problem: Literature Review. **PROMET - Traffic & Transportation**, v. 27, n. 4, p. 345-358, 2015.

HANNAN, M. A. et al. Capacitated vehicle-routing problem model for scheduled solid waste collection and route optimization using PSO algorithm. **Waste Management**, v. 71, p. 31-41, 2018.

HIERHOLZER, C.; WIENER, C. Ueber die möglichkeit einen linienzug ohne wiederholung und thehne unterbrechung zu umfahren. **Mathematische Annalen**, v. 6, n. 1, p. 30-32, 1873.

HOCHBAUM, D. S.; LYU, C.; ORDÓÑEZ, F. Security routing games with multivehicle Chinese postman problem. **Networks**, v. 64, n. 3, p. 181-191, 2014.

IRNICH, S. Solution of real-world postman problems. **European Journal of Operational Research**, v. 190, p. 52–67, 2008.

JACINTO, J. P.; ROSA, R. A.; BANOS, R. S. Heurística para solução do problema da coleta de resíduos sólidos domiciliares (RSD) com base no problema do carteiro chinês capacitado com múltiplas viagens (PCCC-MV). **TRANSPORTES**, v. 22, n. 1, p. 44-55, 2014.

KAUFFMAN, A. *Graphs, dynamic programming and finite games*. Academic Press, New York, 1967.

KELLER, M.T.; TROTTER, W.T. **Applied Combinatorics**. Georgia Institute of Technology: Preliminary Edition, 2015.

KLOIMÜLLNER, C.; PAPAZEK, P.; HU, B.; RAIDL, G.R. A cluster-first route-second approach for balancing bicycle sharing systems. In: **International Conference on Computer Aided Systems Theory**. Springer, Cham, p. 439-446, 2015.

LARSON, R. C.; ODONI, A. R. *Urban operations research*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

LENIS, S. A.; RIVERA, J. C. A Metaheuristic Approach for the Cumulative Capacitated Arc Routing Problem. In: **International Workshop on Experimental and Efficient Algorithms**. Springer, Cham, p. 96-107, 2018.

LI, M.; ZHEN, L.; WANG, S.; LV, W.; QU, X. Unmanned aerial vehicle scheduling problem for traffic monitoring. **Computers & Industrial Engineering**, v. 122, p. 15-23, 2018.

LONGO, H.; DE ARAGÃO, M. P.; UCHOA, E. Solving capacitated arc routing problems using a transformation to the CVRP. **Computers & Operations Research**, v. 33, n. 6, p. 1823-1837, 2006.

LUKMAN, R. K.; CERINŠEK, M.; VIRTIČ, P; HORVAT, B. Improving efficient resource usage and reducing carbon dioxide emissions by optimizing fleet management for winter services. **Journal of cleaner production**, v. 177, p. 1-11, 2018.

MARTINELLI, R.; POGGI, M.; SUBRAMANIAN, A. Improved bounds for large scale capacitated arc routing problem. **Computers & Operations Research**, v. 40, n. 8, p. 2145-2160, 2013.

MASRAN, H.; RAMLI, M. F. Constructive heuristic for the mixed capacitated arc routing problem with multi capacity. In: **Journal of Physics: Conference Series**, IOP Publishing, v. 1317, n. 1, p. 012011, 2019.

MORO, M. F. et al. Otimização de Rotas: um estudo de caso em uma empresa do setor atacadista. **Espacios**, v. 35, n. 7, p. 12-21, 2015.

MORO, M. F.; ANDRADE, F. A.; DOS SANTOS, B. M.; NETO, C. R. P.; BATTISTI, J. F. O problema do carteiro chinês aplicado na otimização das rotas de coleta de resíduos recicláveis: um estudo de caso. **Tecno-Lógica**, v. 22, n. 2, p. 128-135, 2018a.

MORO, M., WEISE, A., DOS REIS, C. C., FLORES, S. Técnicas de pesquisa operacional aplicadas na otimização de rotas de uma rede de lojas de materiais de construção. **Produção em Foco**, v. 8, n. 3, 2018b.

MOURÃO, M. C.; AMADO, L. Heuristic method for a mixed capacitated arc routing problem: A refuse collection application. **European Journal of Operational Research**, v. 160, n. 1, p. 139-153, 2005.

Open Data Commons. 2019. Disponível em: <https://opendatacommons.org/licenses/odbl/>. Acesso em: 12 ago. 2019.

OpenStreetMap. 2019. Disponível em: <www.openstreetmap.org>. Acesso em: 12 ago. 2019.

ORLOFF, C.S. A fundamental problem in vehicle routing. **Networks**, v. 4, p. 5-64, 1974.

PAPADIMITRIOU, C. On the Complexity of Edge Traversing. **Journal of A.C.M.**, v. 23, n. 3, p. 544-554, 1976.

PEARN, W. L.; CHOU, J. B. Improved solutions for the chinese postman problem on mixed networks. **Computers & Operations Research**, v. 26, p. 819-827, 1999.

PEARN, W. L.; LIU, C. M. Algorithms for the Chinese postman problem on mixed networks. **Computers & operations research**, v. 22, n. 5, p. 479-489, 1995.

PORUMBEL, D.; GONÇALVES, G.; ALLAOUI, H.; HSU, T. Iterated local search and column generation to solve arc-routing as a permutation set-covering problem. **European Journal of Operational Research**, v. 256, n. 2, p. 349-367, 2017.

PRAMUDITA, A.; TANIGUCHI, E. Model of debris collection operation after disasters and its application in urban area. **International Journal of Urban Sciences**, v. 18, n. 2, p. 218-243, 2014.

PRINS, C.; LABADI, N.; REGHIOUI, M. Tour splitting algorithms for vehicle routing problems. **International Journal of Production Research**, v. 47, n. 2, p. 507-535, 2009.

PRINS, C.; LACOMME, P.; PRODHON, C. Order-first split-second methods for vehicle routing problems: A review. **Transportation Research Part C: Emerging Technologies**, v. 40, p. 179-200, 2014.

RODRIGUES, A. M.; SOEIRO FERREIRA, J. Waste collection routing-limited multiple landfills and heterogeneous fleet. **Networks**, v. 65, n. 2, p. 155-165, 2015.

SANTOS, L.; COUTINHO-RODRIGUES, J.; CURRENT, J. R. An improved ant colony optimization based algorithm for the capacitated arc routing problem. **Transportation Research Part B: Methodological**, v. 44, n. 2, p. 246-266, 2010.

SAOUB, K. R. *A tour through graph theory*. Taylor & Francis/CRC, 2017.

SCHRIJVER, A. *Theory of linear and integer programming*. Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1998.

SCHRIJVER, A. *Combinatorial Optimization*: Polyhedra and Efficiency. Algorithms and Combinatorics. vol. 24. Springer-Verlag, Berlin, 2003.

SCHUIJBROEK, J.; HAMPSHIRE, R. C.; VAN HOEVE, W. J. Inventory rebalancing and vehicle routing in bike sharing systems. **European Journal of Operational Research**, v. 257, n. 3, p. 992-1004, 2017.

SCHULTZ, M. et al. Open land cover from OpenStreetMap and remote sensing. **International journal of applied earth observation and geoinformation**, v. 63, p. 206-213, 2017.

SHAFAHI, A.; HAGHANI, A. Generalized maximum benefit multiple chinese postman problem. **Transportation Research Part C: Emerging Technologies**, v. 55, p. 261-272, 2015.

SHERAFAT, H. *Algoritmos Heurísticos de Cobertura de Arcos*. Tese (Doutorado). Programa de Pós-Graduação em Engenharia de Produção – Universidade Federal de Santa Catarina, Florianópolis-SC, 2004.

SHERAFAT, H. Sistema construtor de circuitos e sua aplicação na roteirização de coleta de lixo domiciliar. **Revista GEINTEC**, v. 3, n. 5, p. 329-347, 2013.

SILVA, A. A.; LINS, S. L. S.; XAVIER, A. S. Problema do carteiro chinês para rotas de leituristas de ruas. In: **Anais do XXXIX Encontro Nacional de Engenharia de Produção**, Santos-SP. ABEPRO, 2019a. Disponível em: <http://www.abepro.org.br/biblioteca/TN_STO_292_1648_38972.pdf>. Acesso em: 02 jan. 2020.

SILVA, A. A.; LINS, S. L. S.; XAVIER, A. S. Problema do carteiro chinês direcionado na coleta de lixo urbano. In: **Anais do LI Simpósio Brasileiro de Pesquisa Operacional**, Limeira-SP. SOBRAPO, 2019b. Disponível em: <https://proceedings.science/sbpo-2019/papers/problema-do-carteiro-chines-direcionado-na-coleta-de-lixo-urbano?lang=pt-br> Acesso em: 02 jan. 2020.

SOUZA, P. S.; GONÇALVES, N. A. L.; CURI, R. C. Gestão dos resíduos sólidos no Município de Queimadas (Estado da Paraíba, Nordeste do Brasil) segundo a Política Nacional de Resíduos Sólidos. **Rev. Bras. Gest. Ambient. Sust**., v. 5, n. 10, p. 739-752, 2018.

STERN, H. I.; GERTSBAKH, I. B. Using deficit functions for aircraft fleet routing. **Operations Research Perspectives**, v. 6, p. 100104, 2019.

STERN, H. I; DROR, M. Routing electric meter readers. **Computers & Operations Research**, v. 6, n. 4, p. 209-223, 1979.

TING, C. J.; TSAI, H. S. Ant Colony Optimization with Path Relinking for the Capacitated Arc Routing Problem. **Asian Transport Studies**, v. 5, n. 2, p. 362-377, 2018.

TIRKOLAEE, E. B.; MAHDAVI, I.; ESFAHANI, M. M. S. A robust periodic capacitated arc routing problem for urban waste collection considering drivers and crew's working time. **Waste Management**, v. 76, p. 138-146, 2018.

TUCKER, A. **Applied combinatorics**. Wiley, 6 ed. New York, 2012.

ULUSOY, G. The fleet size and mix problem for capacitated arc routing, **European Journal Operation Research**, n. 22, p. 329–337, 1985.

USBERTI, F. L.; FRANÇA, P. M.; FRANÇA, A. L. M. Roteamento de leituristas: Um problema NP-difícil. In: **Anais do XL Simpósio Brasileiro de Pesquisa Operacional**. João Pessoa-PB, 2008.

VAN BEVERN, R.; KOMUSIEWICZ, C.; SORGE, M. A parameterized approximation algorithm for the mixed and windy capacitated arc routing problem: Theory and experiments. **Networks**, v. 70, n. 3, p. 262-278, 2017.

VECCHI, T. P. B. et al. Uma abordagem sequencial para otimização de rotas dos caminhões de coleta de resíduos sólidos. *Congresso de Métodos Numéricos em Engenharia*, 2015. Disponível em: <http://www.dem.ist.utl.pt/cmn2015/html/CD-Proceedings/PDF/Papers/CMN_2015_submission_255.pdf>. Acesso em: 05 jan. 2019.

VECCHI, T. P. B. et al. A sequential approach for the optimization of truck routes for solid waste collection. **Process Safety and Environmental Protection**, v. 102, p. 238-250, 2016.

VIDAL, T. Node, edge, arc routing and turn penalties: Multiple problems–one neighborhood extension. **Operations Research**, v. 65, n. 4, p. 992-1010, 2017.

WANG, H. F.; WEN, Y. P. Time-constrained Chinese postman problems. **Computers & Mathematics with applications**, v. 44, n. 3-4, p. 375-387, 2002.

WANG, J.; TANG, K.; LOZANO, J. A.; YAO, X. Estimation of the distribution algorithm with a stochastic local search for uncertain capacitated arc routing problems. **IEEE Transactions on Evolutionary Computation**, v. 20, n. 1, p. 96-109, 2015.

WILLEMSE, E. J.; JOUBERT, J. W. Constructive heuristics for the mixed capacity arc routing problem under time restrictions with intermediate facilities. **Computers & Operations Research**, v. 68, p. 30-62, 2016.

WØHLK, S.; LAPORTE, G. A fast heuristic for large-scale capacitated arc routing problems. **Journal of the Operational Research Society**, v. 68, n. 12, p. 1877-1887, 2018.

WØHLK, S.; LAPORTE, G. A districting-based heuristic for the coordinated capacitated arc routing problem. **Computers & Operations Research**, v. 111, p. 271-284, 2019.

YAO, T. et al. Memetic algorithm with adaptive local search for Capacitated Arc Routing Problem. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Yokohama, p. 836-841, 2017.

YAOYUENYONG, K.; CHARNSETHIKUL, P.; CHANKONG, V. A heuristic algorithm for the mixed chinese postman problem. **Optimization and Engineering**, v. 3, n. 2, p. 157-187, 2002.

YILMAZ, M.; ÇODUR, M. K.; YILMAZ, H. Chinese postman problem approach for a large-scale conventional rail network in Turkey. **Tehnicki Vjesnik-Technical Gazette**, v. 24, n. 5, p. 1471-1477, 2017.

YU, M.; JIN, X.; ZHANG, Z.; QIN, H.; LAI, Q. The split-delivery mixed capacitated arc-routing problem: Applications and a forest-based tabu search approach. **Transportation Research Part E: Logistics and Transportation Review**, v. 132, p. 141-162, 2019.