UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Botucatu

FCA
FACULDADE DE CIÊNCIAS AGRONÔMICAS
UNESP | BOTUCATU

CARLOS MANUEL OCAMPO ORTIZ

OTIMIZAÇÃO DO PROCESSO DE COLETA DE RESÍDUOS PARA PRODUÇÃO DE BIOGÁS

Botucatu

2021

CARLOS MANUEL OCAMPO ORTIZ

OTIMIZAÇÃO DO PROCESSO DE COLETA DE RESÍDUOS PARA PRODUÇÃO DE BIOGÁS

Dissertação apresentada à Faculdade de Ciências Agronômicas da Unesp Campus de Botucatu, para obtenção do título de Mestre em Agronomia (Energia na Agricultura).

Orientadora: Dra. Helenice de O. Florentino Silva.

Botucatu

2021

O77o

Ortiz, Carlos Manuel Ocampo
Otimização do processo de coleta de resíduos para produção de biogás / Carlos Manuel Ocampo Ortiz. -- Botucatu, 2021
87 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências Agronômicas, Botucatu
Orientadora: Helenice de O. Florentino Silva

1. Otimização. 2. Biodigestor. 3. Roteamento de veículos. 4. Minimização de custos. 5. Heurística. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências Agronômicas, Botucatu. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Botucatu

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: OTIMIZAÇÃO DO PROCESSO DE COLETA DE RESÍDUOS PARA PRODUÇÃO DE BIOGÁS

AUTOR: CARLOS MANUEL OCAMPO ORTIZ

ORIENTADORA: HELENICE DE OLIVEIRA FLORENTINO SILVA

Aprovado como parte das exigências para obtenção do Título de Mestre em AGRONOMIA (ENERGIA NA AGRICULTURA), pela Comissão Examinadora:

Profª. Drª. HELENICE DE OLIVEIRA FLORENTINO SILVA (Participaçao Virtual)
Departamento de Bioestatistica Biologia Vegetal Parasitologia e Zoologia / Instituto de Biociencias de Botucatu UNESP

P/

Prof.ª Dr.ª MAURA SEIKO TSUTSUI ESPERANCINI (Participaçao Virtual)
Engenharia Rural e Socioeconomia / Faculdade de Ciências Agronômicas de Botucatu - UNESP

P/

Profa. Dra. SILVELY NOGUEIRA DE ALMEIDA SALOMAO NEIA (Participaçao Virtual)
Departamento de Estatística / Faculdade de Ciencias e Tecnologia de Presidente Prudente

Botucatu, 12 de novembro de 2021

Ao Juan Pablo,

meu Sol nesses dias escuros,

dedico.

AGRADECIMENTOS

Aos meus queridos pais e irmãos pelo apoio incondicional.

À Prof. Dra. Helenice de O. Florentino Silva, pela orientação, ensinamentos, paciência e exemplo.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela bolsa de estudos concedida – Código de Financiamento 001.

Ao CNPQ – Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela bolsa de estudos concedida.

“A agricultura é a profissão própria do sábio, a mais adequada aos simples e a ocupação mais digna para todos os homens livres.”

GIRALDO, M. La agricultura es la profesión del sabio, la más adecuada al sencillo y la ocupación más digna para todo hombre libre. **Revista Facultad Nacional de Agronomía**, v. 74, n. 2, 2021.

RESUMO

Os biodigestores são instalações onde os resíduos orgânicos são fermentados, produzindo o biogás e o biofertilizante. O biogás é rico em metano, o qual é usado principalmente para geração de energia. Para a produção do biogás em grande escala é necessário recolher o resíduo a ser usado em diferentes pontos. Nas áreas rurais, os pontos de coleta de resíduos estão em diferentes fazendas. Este processo de coleta requer um eficiente planejamento, pois o excesso de viagens em busca de resíduos nas fazendas podem gerar custos econômicos e ambientais indesejáveis, devido ao intenso consumo de combustível. Neste contexto, este trabalho propõe um modelo de otimização visando determinar rotas para os caminhões de coleta dos resíduos que minimizem o custo de transporte. Propõe-se também o uso de heurísticas para a resolução do modelo matemático. Tais técnicas foram implementadas computacionalmente e testadas utilizando diferentes cenários retratando a realidade. Os resultados apresentados mostraram que as metodologias propostas têm potencial para serem utilizadas no processo de tomada de decisões de empresas produtoras de biogás.

Palavras-chave: biodigestores; resíduos orgânicos; roteamento de veículos*;* heurística; minimização de custos.

ABSTRACT

Biodigesters are facilities where organic waste is fermented, producing biogas and biofertilizer. Biogas is rich in methane, which is mainly used for power generation. For large-scale biogas production it is necessary to collect the waste to be used at different points. In rural areas, the waste collection points are on different farms. This collection process requires efficient planning, as excessive travel in search of residues on farms can generate undesirable economic and environmental costs, due to the intense consumption of fuel. In this context, this work proposes an optimization model aiming to determine routes for waste collection trucks that minimize the cost of this transport. The use of heuristics to solve the mathematical model is proposed as well. Such techniques were computationally implemented and tested using different scenarios portraying reality. The results presented showed that the proposed methodologies have the potential to be used in the decision-making process of biogas producing companies.

Keywords: biodigesters; organic waste; vehicle routing; heuristics; cost minimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AGNU	Assembleia Geral das Nações Unidas
CPP	Chinese Postman Problem
CPR	Centro de Processamento de Resíduos
CTCN	Climate Technology Centre & Network
CVRP	Capacitated Vehicle Routing Problem
GRP	General Routing Problem
P	Deterministic Polynomial Time
NP	Non Deterministic Polynomial Time
TSP	Travelling Salesman Problem
TSPTW	Travelling Salesman Problem with Time Windows
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

SUMÁRIO

1 INTRODUÇÃO

Autores como o Christopher (2011) e o Fleishmann (2001) concordam que as indústrias necessitam se adequar ao desafio de preservação do meio ambiente e que uma das ações necessárias é fazer um descarte correto de seus resíduos. Quando se trata de resíduos orgânicos, um dos meios mais apropriados é a sua disponibilização para utilização em biodigestores, visando a produção de gás metano (CH_4), que pode ser aproveitado na geração de energia. Em áreas rurais, tais resíduos orgânicos podem ser utilizados em substituição a outras biomassas sólidas, como lenha e carvão, ou a outras fontes de energia não renováveis, como o petróleo.

Clemens et al. (2018) afirmam que a energia utilizada para fins de cozimento e aquecimento, nas áreas rurais, é gerada principalmente a partir de lenha e carvão. Como alternativa, o biogás pode substituir essas fontes de energia onde houver material orgânico suficiente. Alguns benefícios de sua implementação incluem uma melhoria nas condições de saúde e vida das famílias rurais: maior rendimento da colheita, redução no consumo ou despesa de combustível, e conveniência e limpeza na preparação de comidas; uma redução no desmatamento e nas emissões de gases de efeito estufa e um aumento nas oportunidades de emprego rural. Além disso, como afirmam Aguilar et al. (2006), os efluentes oriundos da biodigestão possuem alta concentração de nutrientes, baixo teor de patógenos e são praticamente isentos de sementes rasteiras, sendo assim de alta importância para fins agrícolas. Esta afirmação foi comprovada por Villas Bôas et al. (2004) e Van Hiep et al. (2006), que expõem os efeitos dos compostos orgânicos na produção de alface e a melhoria na quantidade de biomassa de espinafre produzida, conforme aumenta o nível de efluentes do biodigestor aos quais a cultura está exposta, respetivamente.

Segundo Lansing et al. (2008), as tecnologias de biodigestores e a atividade de microrganismos como bactérias, algas, fungos, plantas e animais tem sido amplamente estudadas visando diminuir os impactos das atividades agrícolas sobre o meio ambiente, visto que as águas residuais são lançadas em córregos, rios e lagos. Para mitigar esses efeitos, várias abordagens têm sido estudadas, incluindo algumas que integram biodigestores construído com plástico de baixo custo, canais de plantas aquáticas, lagoas de policultura de peixes e plantações, e codigestão com sustâncias ricas em resíduos orgânicos, como no trabalho apresentado por Lansing et al. (2010).

Em relação às contribuições energéticas, Dornelas et al. (2017) destacam uma alta eficiência no sistema de biodigestão anaeróbia a partir de dejetos de aves e um aumento substancial na produção de biogás pelo reaproveitamento desses resíduos. Outras aplicações, como a descrita por Chaudhari et al. (2007), mostram que o lodo seco, que vem da coagulação e floculação do efluente do biodigestor em uma destilaria à base de melaço de cana-de-açúcar, possui um valor calorífico médio, podendo ser usado como combustível em um biorreator, forno ou incinerador. Além disso, metais como Al, Fe, Na, K, P e Mg estão presentes nesse resíduo e permanecerão nas cinzas após a sua combustão. Estes macronutrientes podem ser misturados ao esterco e usados como composto.

Apesar da grande importância do aproveitamento de resíduos para descontaminação do meio ambiente e geração de energia limpa e renovável, ainda existem grandes desafios de logística de transporte destes resíduos, principalmente das agroindústrias para os centros de processamento, uma vez que esta é muito complexa e envolve um grande número de rotas, de veículos, pontos de captação e de entrega do resíduo. Quando este processo logístico é mal planejado, incrementa-se o consumo de combustível, igualmente o custo do produto final e surgem problemas de atendimento da demanda de energia. Neste contexto, é de suma importância a investigação de técnicas matemáticas buscando a otimização da logística de recolhimento dos resíduos nas agroindústrias e entrega nos centros de localização dos biodigestores destinados a produção de biogás para geração de energia, isto minimiza o custo econômico, e indiretamente o custo ambiental, deste processo. Alinhado a este fato e seguindo as diretrizes da AGNU (2015) na luta pela sustentabilidade, este trabalho investiga técnicas matemáticas de logística inversa, auxiliando assim no processo de geração de energia renovável, suportando a reciclagem de resíduos industriais e promovendo diminuição do impacto ambiental deste.

O texto está dividido como segue: O Capítulo 2 é dedicado a apresentação dos conceitos e definições necessários para entendimento e contextualização do problema abordado; o Capítulo 3 apresenta os materiais e métodos utilizados; o Capítulo 4 apresenta e discute os resultados alcançados. Finalmente, o Capítulo 5 expõe as conclusões obtidas e sugere possíveis trabalhos futuros derivados desta pesquisa.

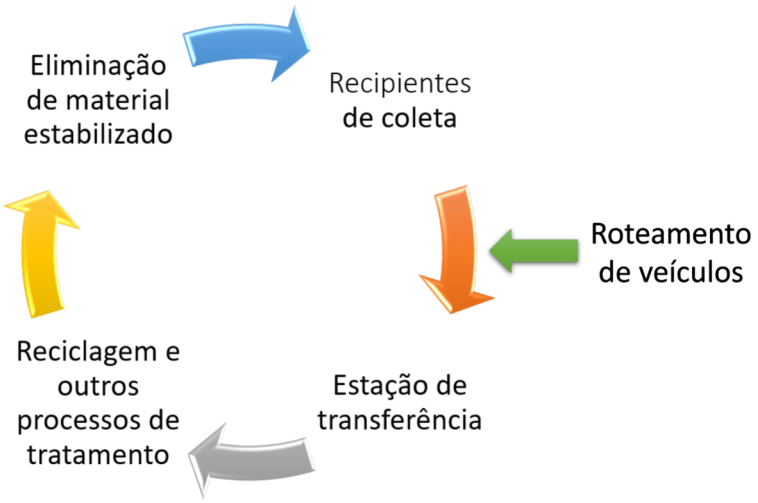
2 REVISÃO DE LITERATURA

Neste capítulo são apresentados, levando em consideração estudos feitos na área desta pesquisa, os conceitos necessários para a compreensão dos temas mais relevantes para esta dissertação. Nesse sentido, primeiro é introduzida a questão do manejo de resíduos, em seguida, suas opções de aproveitamento em biodigestores como solução para o seu descarte e, posteriormente, o papel fundamental da otimização da coleta de resíduos nesse processo. Chegado este ponto, definições matemáticas e conceitos-chave são apresentados para sustentar o núcleo deste trabalho: o roteamento de veículos.

2.1 Manejo de resíduos

Autores como Ripa et al. (2017) e Wasserman et al. (2005) mostram a importância da gestão de resíduos na sociedade atual, quer a partir da análise do ciclo de vida dos aterros com base em dados específicos das suas localizações, quer estabelecendo a sua relevância em função das diferentes estratégias implementadas na gestão. Outros estudos mostram que o desenvolvimento de sistemas efetivos e ambientalmente sustentáveis para o manejo de resíduos tornou-se um grande desafio na agroindústria, pois envolve o processo de geração, recolhimento, transporte, tratamento, recuperação de valor e eliminação subsequente. Um planejamento inadequado de algum desses processos pode aumentar o custo operacional e a poluição ambiental. Somente os processos de recolhimento e transporte, custam aproximadamente 60%-80% do custo total de manejo dos resíduos (SULEMANA et al. 2018). A redução de custos com respeito aos já mencionados processos, é essencial para alcançar um manejo de resíduos sólidos sustentável nas economias em desenvolvimento. Portanto, é preciso planejar o sistema de recolhimento de resíduos sólidos, o qual envolve a coleta e o transporte do resíduo (veja a Figura 1), de forma eficiente e eficaz; com esse propósito são usadas técnicas como de análises de sistemas e otimização de operações.

Figura 1 – Processo de coleta de resíduos sólidos.



Fonte: Sulemana et al. (2018, p. 4). Traduzida pelo autor.

Naturalmente, o processo de roteamento de veículos é o mais importante e custoso no manejo de resíduos devido à intensidade de trabalho e o uso maciço de caminhões. O processo de coleta de resíduos pode ser dificultado pela falta de financiamento, falta de vontade, prioridade e compromisso político; rede de estradas e planejamento de desenvolvimento inadequado; fatores técnicos inadequados e, por último mas não menos importante, falta de planejamento operacional e roteamento de veículos (SULEMANA et al. 2018), aspecto com o qual esta pesquisa pretende contribuir.

2.2 Biodigestores

Segundo o *Climate Technology Centre & Network (*CTCN, 2016), um biodigestor é um sistema que usa resíduos orgânicos, por exemplo fezes animais e humanas, para produzir adubo e biogás. Este é composto de um recipiente hermético, no qual se fermentam os resíduos diluídos em água. Esta fermentação ocorre devido à presença de um consórcio de diferentes tipos de bactérias nesta mistura. O processo de fermentação é anaeróbico, isto é, ocorre na ausência de oxigênio, e desta decomposição é gerado o biogás, gás rico em metano e dióxido de carbono. Além do biogás, outro produto muito importante da biodigestão é o biofertilizante, substrato remanescente no biodigestor, livre de patógenos, rico em nitrogênio, fósforo e potássio, podendo ser utilizado como fonte de nutriente para o solo das lavouras locais.

Conforme afirma Mukherjee et al. (2015), o biogás gerado nos biodigestores anaeróbios consiste em metano (50% a 80%), dióxido de carbono (50 a 20%) e algumas pequenas quantidades de outros gases. O biogás pode ser aproveitado para geração de calor e energia, como combustíveis para transporte e como intermediários químicos. Apesar das vantagens mencionadas, um dos obstáculos enfrentados por esta tecnologia é a escala de operação necessária. Uma alternativa para esse inconveniente é a utilização de biodigestores centralizados, ou comunitários, que permita a participação conjunta de diferentes fazendas, obtendo o insumo necessário à operação e a rentabilidade desejada (CANTRELL et al. 2008), mesmo assim eles apresentam um custo adicional pela coleta e transporte dos resíduos, daqui a importância deste trabalho.

Para o sucesso da criação e operação destes biodigestores centralizados, é necessário ter um bom planejamento logístico em todo o processo, de forma que os resíduos sejam corretamente recolhidos e disponibilizados pelas fazendas; após coletada, esta matéria prima é transferida, em caminhões, para a central de biodigestão. Outro fator importante neste planejamento é a trajetória dos caminhões que transferem a matéria prima para a central de biodigestão, pois rotas mais curtas demandam menor consumo de combustível e portanto têm menor custo econômico e ambiental.

O planejamento de todo este processo é bastante complexo e necessita de ferramentas matemáticas e computacionais para se suportar. Desta forma, a otimização matemática é uma ferramenta útil de análise e tomada de decisões. Tal ferramenta consiste, segundo Edwards et al. (2016), em aplicar técnicas matemáticas a problemas reais e assim obter soluções viáveis, criando modelos que permitem uma avaliação rápida e econômica de alternativas e fornecendo soluções para o problema.

2.3 Modelagem matemática

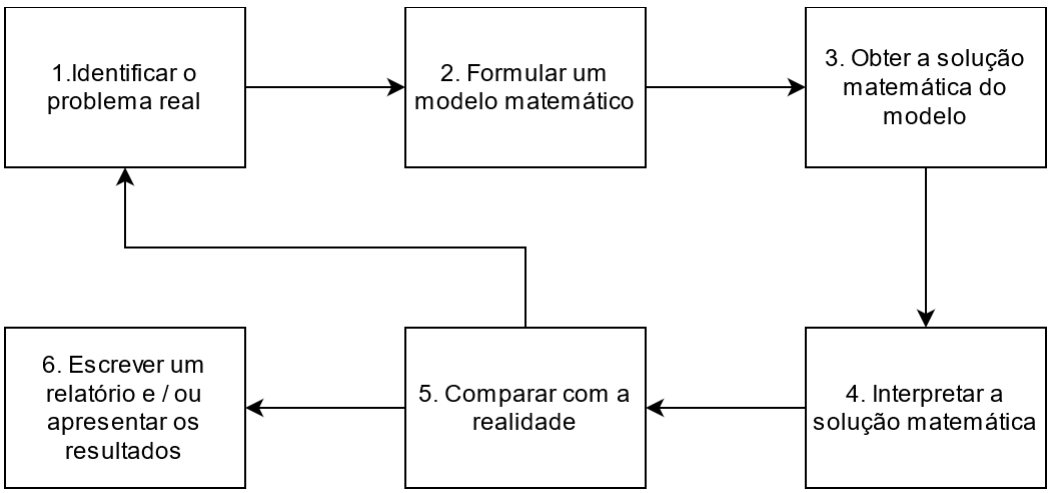
Segundo Pimentel (2014), a modelagem matemática consiste na arte de transformar problemas reais em modelos matemáticos, resolvendo-os e interpretando suas soluções na realidade. A sua importância reside na capacidade que possui de expressar estes problemas com precisão e clareza, viabilizando a sua solução através de métodos computacionais e da correção ou extensão dos modelos existentes. Apesar do exposto, esses modelos exigem simplificações dos problemas

reais, seus resultados são aproximações da realidade e, portanto, devem ser levados em consideração apenas dentro da aproximação feita para a formulação do problema.

Deixando de lado a classificação quanto ao campo de aplicação e a finalidade, os modelos matemáticos podem ser: lineares ou não lineares, se as equações que os descrevem são lineares ou não; estáticos ou dinâmicos, sejam as variações temporárias levadas em consideração ou não; determinísticos ou estocásticos, se os fatores de mudança são ignorados ou não; e discretos ou contínuos, se as variáveis envolvidas forem discretas ou contínuas (KAPUR, 2015).

De acordo com Pimentel (2014), a modelagem matemática de otimização de uma situação ou problema real segue uma determinada ordem (veja a Figura 2): primeiro, o problema real a ser resolvido é estudado e as variáveis são identificadas, a seguir os objetivos são identificados e o modelo é formulado considerando as restrições do problema real. As hipóteses são formuladas por observação dos fatos, comparações, experiências pessoais, dedução lógica ou técnica. Se necessário, devido à sua excessiva complexidade, o problema pode ser simplificado, sem chegar ao ponto de perder informações essenciais. Posteriormente, o modelo é resolvido e os resultados são discutidos. Por fim, ocorre a validação do modelo: nesta etapa é posto à prova e analisado para avaliar o seu grau de aproximação à realidade. Dependendo da validação, é possível fazer alterações no modelo e reiniciar o processo.

Figura 2 – Fluxograma do processo de modelagem matemática.



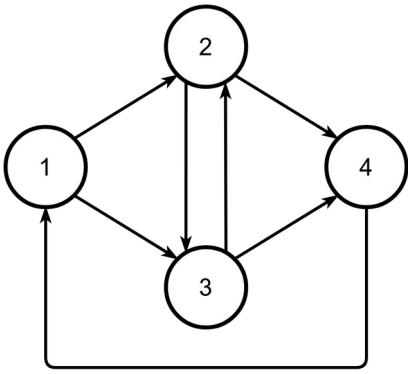
Fonte: Edwards et al. (2016, p. 44). Traduzida pelo autor.

Para a modelagem de um problema real, é necessário a aplicação de conceitos matemáticos. Considerando o foco desta pesquisa, a seguir são apresentados alguns conceitos básicos da teoria de redes, essenciais para tratar do problema de roteamento de veículos.

2.4 Grafos

Os problemas envolvendo transporte quase sempre podem ser representados em forma de grafos. De acordo com Bazaraa et al. (2009), um grafo é uma *rede direcionada* G que consiste em um conjunto finito de m *nós* (pontos) $N=\{1,2,\cdots,m\}$ e um conjunto de n *arcos direcionados (linhas)* $S=\{(i,j),(k,l),\cdots(s,t)\}$ unindo pares de nós em N. O arco (i,j) é considerado *incidente* com os nós i e j e sua direção vai do nó i para o nó j. A Figura 3 apresenta um grafo com 4 nós e 7 arcos.

Figura 3 – Exemplo de uma grafo.

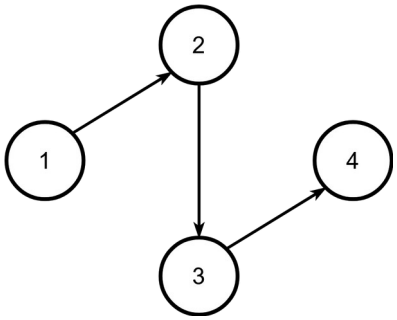


Fonte: Bazaraa et al. (2009, p. 455). Adaptada pelo autor.

A fim de esclarecer a notação, é necessário apresentar uma distinção entre quatro conceitos frequentemente tratados na teoria dos grafos:

Um *caminho* (veja a Figura 4) do nó i_0 até i_p, é uma sequência de arcos $P=\{(i_0,i_1),(i_1,i_2),\cdots,(i_{p-1},i_p)\}$ em que o nó inicial de cada arco coincide com o nó terminal do arco anterior na sequência, sendo $i_0,\cdots,i_p$ nós distintos. Assim, cada arco no caminho é direcionado “para” i_p e “para longe de” $i_{0.}$

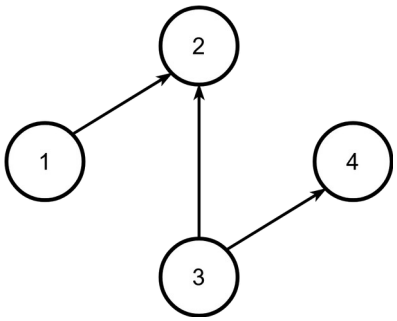
Figura 4 – Exemplo de um caminho.



Fonte: Bazaraa et al. (2009, p. 455). Adaptada pelo autor.

Uma *cadeia,* exemplificada na Figura 5, é uma estrutura semelhante a um caminho, exceto que nem todos os arcos são necessariamente direcionados para o nó i_p.

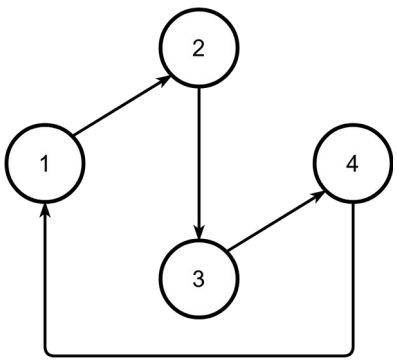
Figura 5 – Exemplo de uma cadeia.



Fonte: Bazaraa et al. (2009, p. 455). Adaptada pelo autor.

Um *circuito* é um caminho de algum nó i_0 até i_p mais o arco (i_p, i_0). Assim, um circuito é um *caminho fechado* (veja a Figura 6).

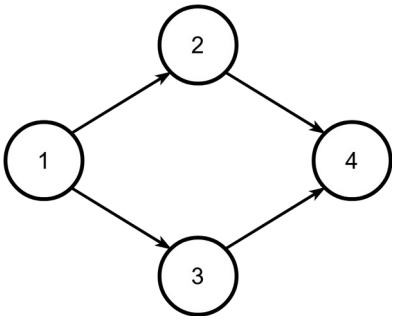
Figura 6 – Exemplo de um circuito.



Fonte: Bazaraa et al. (2009, p. 455). Adaptada pelo autor.

Da mesma forma, um *ciclo*, ilustrado na Figura 7, é uma *cadeia fechada*.

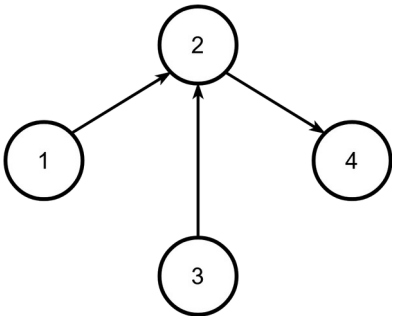
Figura 7 – Exemplo de um ciclo.



Fonte: Bazaraa et al. (2009, p. 455). Adaptada pelo autor.

Uma árvore, representada na Figura 8, é um grafo conectado sem ciclos.

Figura 8 – Exemplo de uma árvore.



Fonte: Bazaraa et al. (2009, p. 455). Adaptada pelo autor.

2.5 Roteamento de veículos

Um problema importante de otimização, e de difícil solução, é o problema de roteamento de veículos. Tal problema é geralmente classificado de acordo com o conjunto de paradas necessárias em uma rede de nós (fazendas, cidades etc.) que devem ser visitados, ou de um conjunto de arcos (estradas, ruas, caminhos, etc.) em uma rede que necessariamente devem ser percorridos. O problema de percorrer todos os arcos (estradas, no caso deste trabalho) de uma rede com uma rota mínima é o bem conhecido Problema do Carteiro Chinês (C*hinese Postman Problem* ou *CPP*). O problema de percorrer cada arco (pelo menos uma vez) em um subconjunto necessário de todos os arcos de uma rede a um custo mínimo é chamado de Problema do Carteiro Rural (*Rural Postman Problem* ou *RPP*). O problema de percorrer todos os nós de uma rede com uma rota mínima é conhecido como

Problema do Caixeiro Viajante (*Travelling Salesman Problem* ou *TSP*). O problema geral de roteamento (*General Routing Problem* ou *GRP*) consiste em encontrar a rota de custo mínimo que atravessa cada arco em um subconjunto necessário de arcos da rede e que visita cada nó em um subconjunto necessário de nós da rede. O *TSP*, *CPP* e *RPP* são todos casos especiais do *GRP* (ORLOFF, 1974).

A Tabela 1 mostra diferentes pesquisas que utilizam técnicas de otimização matemática e estão focadas no roteamento de veículos, visando a otimização de rotas para coleta de resíduos, levando a resultados quantificáveis e favoráveis.

Tabela 1 – Estudos sobre o problema de roteamento de veículos baseados em otimização matemática (lista não exaustiva).

Autor (es)	Foco	Abordagem / Resultado
Clark et al. (1975)	Análise das operações de gestão de resíduos sólidos.	Descreve a aplicação do modelo de simulação para os problemas complexos do sistema de gestão de resíduos sólidos que reduziu o orçamento anual de US$ 14,8 milhões em 1970 para US$ 8,8 milhões em 1972 e força de trabalho de 1640 a 850.
Ronen et al. (1983)	Melhoria de um sistema de coleta de resíduos sólidos.	Usa um método heurístico para a análise e modificação de rotas de coleta de resíduos, e implementação de rotas propostas. Possibilitou reduzir a distância total percorrida em 18,7%.
Sonesson (2000)	Modelagem de coleta de resíduos para calcular o consumo de combustível e tempo.	Utiliza estatísticas comuns a modelo para fazer estimativas de produção de energia e para redução de tempo trabalhado.
Awad et al. (2001)	Desenvolvimento de um procedimento simplificado para o roteamento e para coleta de resíduos sólidos.	Resolve o problema do carteiro chinês e problema do caixeiro viajante usando simulação de Monte Carlo, algoritmo heurístico e algoritmo heurístico modificado para sugerir sistema de roteamento adequado.
Aringhieri et al. (2004)	Um problema de roteamento de veículos assimétrico que surge na coleta e eliminação de resíduos especiais.	Fornece um modelo gráfico baseado em uma formulação de roteamento de veículos assimétrica e discute algoritmos heurísticos para coleta de resíduos recicláveis volumosos. Economizou tempo de viagem e número de veículos utilizados ou necessários.
Ghiani et al. (2005)	Coleta de resíduos: solução do problema de roteamento em arco.	Usa o problema de roteamento em arco em problemas reais para redução do custo total dos processos de transporte envolvidos.
Agha (2006)	Roteamento ótimo de veículos para coleta de resíduos sólidos urbanos.	Otimiza sistema de roteamento usando o modelo de programação inteira mista. Resultou na redução de 23,47% da distância percorrida, economizando cerca de US$ 1.140,00 por mês.

Dell et al. (2006)	Abordagem *Branch-and-price* para o problema de roteamento de veículos com a distribuição e coleta simultânea.	Apresenta algoritmos *branch-and-price* para a solução exata do problema de roteamento de veículos com entrega e coleta simultânea, sem qualquer restrição adicional.
Johansson (2006)	Efeito da programação dinâmica e roteamento no sistema de gestão de resíduos sólidos	Usa modelagem analítica e simulação de eventos discretos para avaliar diferentes programação e políticas de roteamento utilizando dados em tempo real.
von Poser et al. (2006)	Roteamento ideal para a coleta de resíduos sólidos.	Usa roteamento em nó ou problema do caixeiro viajante para desenvolver uma metodologia baseada em algoritmo genético. Os resultados indicam que o algoritmo genético produz significativamente a solução de menor distância (rota de custo mínimo).
Cordeau et al. (2007)	Roteamento de veículos.	Apresenta uma visão abrangente dos algoritmos disponíveis exatos e heurísticos para a VRP (*Vehicle Routing Problem*), a maioria dos quais foram adaptados para resolver outras variantes.
Reinhardt et al. (2011)	Encaminhamento e agendamento de problemas.	Usos de programação dinâmica para caminhos mais curtos com várias restrições, *branch-and-cut* para o transporte linear, recozimento simulado para o transporte de passageiros atendidos em aeroportos, *branch-cut-and-price* para o roteamento de veículos com janelas de tempo e custos fixos limites.
Gyamfi et al. (2012)	Ordenação sequencial de rotas de caminhões para coleta de lixo eficiente.	Revela excelente desempenho do algoritmo de otimização colônia de formigas em termos de eficiência: redução de custo total em 35%.
Otoo (2012)	Problema de roteamento com capacidade: coleta de resíduos sólidos.	Usa o caminho emparelhado mais curto e pontos de coleta particionados com base na capacidade de veículos e a metaheurística colônia de formigas para encontrar a rota mínima e melhorou distância total em 40%.
Naninja (2013)	Otimização do custo de transporte de resíduos sólidos.	Reduz número de viagens de veículos para minimizar os custos através da formulação de um problema de programação linear inteira.
Das et al. (2015)	Otimização de rotas para coleta de resíduos municipais.	Propõe um sistema de coleta e transporte ótimo que incide sobre o problema de minimizar o comprimento de cada percurso de coleta e transporte através de uma solução heurística usando programação inteira mista.
Han et al. (2015)	Revisão sobre o problema de roteamento de veículos para coleta de resíduos	Analisa a contribuição importante sobre veículo de coleta de resíduos.
Toro et al. (2016)	Revisão sobre o problema de roteamento de veículos no transporte verde.	Propõe a interação entre as variantes dos problemas clássicos de roteamento e os efeitos ambientais das suas operações, conhecidas na literatura como *Green-VRP.*

Fonte: SULEMANA et al. (2018, p. 6).

Para finalizar, conforme comprovado por Archetti (2009), o Problema de Roteamento de Veículos (*VRP*) pertence à categoria *NP-hard*, ou seja, não há algoritmo de resolução em tempo polinomial ou, nas palavras de Cao (2012), “um grande *VRP* não pode ser resolvido de forma ótima dentro de um tempo computacional aceitável”. Além disso, de acordo com Solomon (1987) e Cordeau (2002), o *VRPTW* também é *NP-hard*, uma vez que generaliza o problema de roteamento de veículos com capacidade (*CVRP*) e envolve o problema do caixeiro viajante com janelas de tempo (*TSPTW*). Mesmo encontrar uma solução viável para o *VRPTW* quando o número de veículos é fixo é um problema *NP-completo*.

A partir destes estudos preliminares, teve-se um direcionamento para o desenvolvimento de uma metodologia de resolução do problema de otimização do processo de coleta de resíduos para produção de biogás. Esta metodologia está apresentada no capítulo a seguir e consiste na elaboração de um modelo matemático que descreva o problema abordado e os processos de resolução deste.

3 MATERIAL E MÉTODOS

A metodologia de modelagem matemática de um problema real consiste de seis passos sequenciais, os quais são descritos a seguir:

- O primeiro passo para modelagem matemática de um problema real (ou processo) é a completa descrição e entendimento deste, reforçando as características importantes. Se o problema envolver otimização de processos, deve-se incluir as restrições envolvidas e o objetivo (ou objetivos) a ser alcançado.
- O segundo passo é a identificação das variáveis e parâmetros que serão usados no modelo.
- O terceiro passo consiste na descrição do problema real em expressões matemáticas, utilizando as variáveis e parâmetros identificados (formulação do modelo matemático). Muitas das complexidades dos problemas devem ser simplificadas para que o modelo formulado tenha condições de ser resolvido de forma analítica ou computacional.
- Com o modelo formulado, o quarto passo consiste da identificação de métodos matemáticos para resolução deste.
- O quinto passo está relacionado com a resolução analítica ou computacional do modelo, utilizando os métodos identificados no quarto passo. Deve-se também obter valores para os parâmetros matemáticos abordados no modelo.
- O sexto passo consiste na análise e discussão dos resultados alcançados no quinto passo, assim como na integração destes resultados com a solução do problema real e uma análise do modelo como ferramenta para auxílio na resolução do problema abordado. Caso o modelo seja aprovado como a ferramenta desejada, o processo é finalizado. Caso o modelo seja inviável para resolução do problema real, adaptações terão que ser realizadas para resolver as deficiências do modelo, retornando ao terceiro passo.

A metodologia descrita foi aplicada no problema objeto deste trabalho. Os passos de um até quatro são discutidos a seguir. Os passos cinco e seis serão apresentados no capítulo seguinte.

3.1 Descrição do problema

O problema de roteamento abordado neste trabalho consiste em otimizar a logística da coleta de resíduos orgânicos gerados em diferentes fazendas e a entrega destes nos centros de biodigestão, de forma a atender a demanda local de biogás com um mínimo custo econômico e ambiental, ou seja, reduzindo o consumo de combustível e o custo operacional.

Em geral, uma frota de caminhões parte das unidades biodigestoras ou centros de processamento de resíduos (CPR) e percorre os pontos de coleta; uma vez totalmente carregados, estes veículos retornam aos seus respectivos pontos de partida. O objetivo é achar uma rota de custo mínimo para os caminhões percorrerem, enquanto os requerimentos do(s) CPR são atingidos. Esta rota otimizada oferece um caminho com menor distância percorrida, refletindo em um menor consumo de combustível e portanto, menor custo.

Portanto, o problema aqui abordado consiste em: Dado um CPR e os resíduos gerados em n fazendas (pontos de coleta), em que o CPR tem uma demanda d de resíduos e cada ponto de coleta i tem uma capacidade l_i de geração de resíduos, $i=1,\ldots,n$. Estes resíduos são coletados por m caminhões de mesma capacidade q. Os custos variáveis (ou distâncias) c_{ij} e os tempos t_{ij} para percorrer qualquer caminho de um local i para um local j são conhecidos. O tempo de carga e descarga de um caminhão em um local i é dado por s_i. Existe um horário específico que cada fazenda disponibiliza o resíduo, que é no intervalo de tempo a_i até b_i, $i=1,\ldots,n$. Isto posto, deseja-se determinar um planejamento das rotas dos caminhões a um custo mínimo.

Na Tabela 2 estão resumidos os parâmetros necessários para a definição do problema de roteamento de veículos com janelas de tempo:

Tabela 2 – Parâmetros do problema de roteamento de veículos.

Parâmetros	
a_i	Horário inicial que o caminhão pode coletar o resíduo no nó i.
b_i	Horário final que o caminhão pode coletar o resíduo no nó i.
c_{ij}	Custo associado à viagem do nó i ao nó j.
d	Demanda de resíduos orgânicos no CPR (massa).
l_i	Disponibilidade de resíduos orgânicos no ponto de coleta i (massa).

L_i^k	Carga acumulada do veículo, na rota k, ao visitar a fazenda i.
m	Número de caminhões disponíveis.
n	Número de pontos de coleta.
q	Capacidade dos caminhões.
s_i	Tempo de serviço no nó i.
t_{ij}	Tempo associado à viagem do nó i ao nó j.
T_i^k	Tempo do início do serviço no nó i, na rota k.

Fonte: O autor.

O sistema de coleta de residuos pode ser visto como um grafo direcionado $G=(V,A)$, em que V é o conjunto de nós e representa os pontos de coleta e o CPR, isto é, o conjunto de nós é composto de $N\cup\{0,n+1\}$, em que N consiste no conjunto de nós que representam pontos de coletas, 0 é o nó de partida e $n+1$ o ponto de chegada do caminhão (CPR). Do mesmo modo, A é o conjunto dos arcos e agrupa as rotas existentes entre os nós. Além disso, a janela de tempo $[a_i,b_i]$, $i\in V$, está associada a cada nó e o custo a cada arco $(i,j)\in A$, assim como um tempo de viagem não negativo t_{ij}. Este último deve satisfazer a condição de viabilidade $a_i+t_{ij}\leq b_j$, onde t_{ij} pode incluir o tempo de serviço no nó i, s_i. O veículo pode chegar antes da abertura da janela de tempo e esperar sem custo até que o serviço seja possível, mas não é permitido chegar após o fechamento da janela de tempo, ou seja, existe um período de tempo pré-definido em que o resíduo fica disponível para coleta em cada fazenda, não sendo possível a retirada deste em outro horário. Os veículos devem deixar o CPR e retornar em um período de tempo estabelecido $[a_0,b_0]$. Finalmente, todos os parámetros, ou seja, q, d, l_i, c_{ij}, t_{ij}, a_i, b_i, e s_i, são assumidos como inteiros não negativos, ademais, os tempos t_{ij} devem ser estritamente positivos.

Para formulação do modelo, considere as variávelis de decisão X_{ij}^k:

$$X_{ij}^k=\begin{cases}1 \text{ se o caminho de } i \text{ para } j \text{ é percorrido na rota } k\\ 0 \text{ em caso contrário}\end{cases}$$

além disso destaca-se a variável $T_i^k, i\in V$, que indica o início do serviço no nó i, dentro da rota k. Deseja-se planejar este sistema de coleta de resíduo com o menor custo possível (o custo está associado ao caminho a ser percorrido), o problema pode ser modelado conforme apresentado a seguir.

$$\min \sum_{k} \sum_{(i,j)\in A} c_{ij} X_{ij}^{k} \tag{1}$$

sujeito a

$$\sum_{j:(0,j)\in A} X_{0j}^{k} = \sum_{i:(i,n+1)\in A} X_{i(n+1)}^{k} \leq m, \forall k \tag{2}$$

$$\sum_{j:(i,j)\in A} X_{ij}^{k} - \sum_{j:(j,i)\in A} X_{ji}^{k} = 0, \forall i \in N, \forall k \tag{3}$$

$$X_{ij}^{k}\left(T_i^{k} + t_{ij} - T_j^{k}\right) \leq 0, \forall (i,j) \in A, \forall k \tag{4}$$

$$\sum_{j:(i,j)\in A} X_{ij}^{k} = 1, \forall i \in N, \forall k \tag{5}$$

$$X_{ij}^{k}\left(L_i^{k} + l_j - L_j^{k}\right) \leq 0, \forall (i,j) \in A, \forall k \tag{6}$$

$$l_i \leq L_i^{k} \leq q, \forall i \in V, \forall k \tag{7}$$

$$a_i \leq T_i^{k} \leq b_i, \forall i \in V, \forall k \tag{8}$$

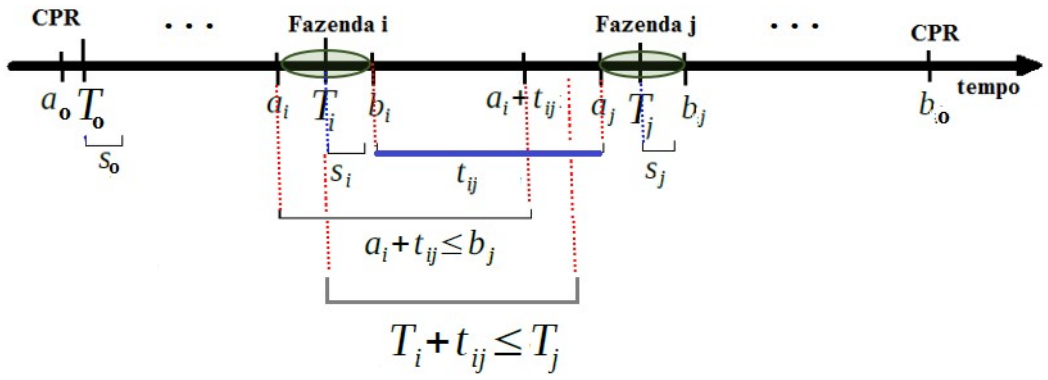
$$\sum_{k} \sum_{j:(i,j)\in A} X_{ij}^{k} l_j \geq d, \forall i \in N \tag{9}$$

$$X_{ij}^{k} \in \{0,1\}, \forall (i,j) \in A, \forall k \tag{10}$$

A função objetivo (1) visa a minimização do custo total da rota, ou seja, dos arcos percorridos. A restrição (2) estabelece o uso de m caminhões que saem do nó 0 e chegam ao nó $n+1$. As restrições (3) definem o fluxo clássico, ou seja, cada ponto de coleta possui uma única rota de entrada e uma única rota de saída associada a ele. As restrições não lineares (4) expressam a compatibilidade necessária entre as variáveis de fluxo e tempo, isto é, na rota k, o caminhão inicia o carregamento do resíduo na fazenda i no momento T_i^{k}. Após o carregamento, tem que percorrer o espaço entre a fazenda i e a fazenda j no tempo t_{ij}, sendo $(T_i^{k}+t_{ij})$ um tempo hábil para iniciar o carregamento do resíduo na fazenda j no momento T_j^{k}, isto é, $(T_i^{k}+t_{ij})$ tem que ser menor ou igual a T_j^{k}. A Figura 9 ilustra estas situações. As restrições (5) garantem que todos os pontos de coleta sejam atendidos. Lembrando que a capacidade dos veículos é denotada como q e a oferta de resíduo no ponto de coleta i é $l_i, i \in V$, a carga acumulada do veículo na rota k ao visitar a fazenda i é definida como L_i^{k}; sem perda de generalidade, pode-se dizer que $l_0 = l_{n+1} = 0$. Uma vez que os arcos (i,j) com $l_i + l_j > q$ são removidos do

conjunto A, isto é, os arcos formados por duas fazendas cujas ofertas combinadas excedem a capacidade de um caminhão são omitidos; as restrições (6) garantem que, ao viajar da fazenda i para a fazenda j, a oferta de resíduo da fazenda i mais a carga acumulada neste ponto não exceda a carga acumulada na fazenda j, ou seja, $L_j^k \leq L_i^k + l_i$. As restrições (7) garantem que a carga acumulada em um ponto de coleta i seja no mínimo l_i e não ultrapasse a capacidade q do caminhão. As restrições (8) garantem que a demanda do CPR seja atendida. As restrições (9) obrigam os caminhões a iniciar o serviço dentro da janela de tempo estabelecida para cada fazenda, e as restrições (10) definem as variáveis de decisão do modelo como binárias.

Figura 9 – Parâmetros temporais envolvidos na modelagem (omite-se o índice de rota).



Fonte: O autor.

O veículo chega no CPR no tempo T_0^k e gasta um tempo s_0 para descarregar o resíduo. Em seguida retorna para sua rota visando uma nova coleta. Chegando em uma fazenda i, inicia sua carga no horário T_i^k (dentro da janela de tempo $[a_i, b_i]$), gastando um tempo s_i para carregar o resíduo e depois parte para a fazenda j. O veículo gasta um tempo t_{ij} para percorrer o caminho de i para j e deve iniciar o novo carregamento em um horário T_j^k na janela de tempo $[a_j, b_j]$. Repetindo este processo em todas os pontos de coleta que passar. Depois de cumprir toda a sua jornada, o veículo deve retornar ao CPR e descarregar o resíduo antes do horário $b_{0.}$

Levando em consideração o modelo anteriormente apresentado, sua formulação e seus parâmetros; a situação aqui tratada se identifica com um caso ampliamente discutido e relatado: o *Vehicle Routing Problem with Time Windows* (*VRPTW*).

3.2 Método de resolução do modelo

Diversas técnicas heurísticas têm sido desenvolvidas para resolver problemas reais e, dentro das possibilidades, fornecer uma abordagem ótima e eficaz. Levando em conta esse fato, Desrochers (1987) afirmou que "apesar do sucesso dos algoritmos de otimização para roteamento de veículos com janelas de tempo, é improvável que eles sejam capazes de resolver problemas de grande escala. Em muitas situações, é necessário se contentar com algoritmos que funcionam rápido, mas podem produzir solução subótima". Diante disso e levando em consideração que ainda não foi encontrado um algoritmo determinístico que resolva exatamente o *VRPTW* (entre outros problemas) em um tempo polinomial, como relatam Cook (2006) e o Clay Mathematics Institute (2021); técnicas puramente heurísticas para resolver o *VRPTW* foram propostas por Solomon (1987), as quais são descritas a seguir.

3.2.1 Heurística de redução de custos

A heurística de redução de custos para solucionar o *VRPTW* foi proposta por Solomon (1987) e está baseada no algoritmo de Clarke e Wright (1964). Segundo Cordeau (2002), essa faz parte da categoria chamada “construção de rota” pois constrói uma possível solução inserindo um cliente não roteado na rota parcial que esteja sendo construída. Embora esse processo possa ser realizado sequencialmente ou em paralelo, a implementação computacional desenvolvida foi estritamente sequencial, ou seja, as rotas foram construídas uma a uma.

O primeiro passo na aplicação desse algoritmo é o cálculo da redução no custo R_{ij}, ao optar por sair de uma fazenda i e ir para uma fazenda j ao invés de retornar ao CPR, dada pela equação (11).

$$R_{ij} = \text{Distância}(0,i,0) + \text{Distância}(0,j,0) - \text{Distância}(0,i,j,0)$$

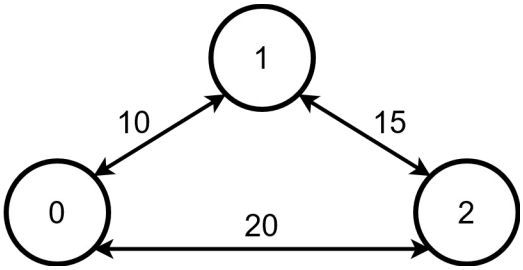
$$R_{ij} = (c_{0i} + c_{i0}) + (c_{0j} + c_{j0}) - (c_{0i} + c_{ij} + c_{j0})$$

$$R_{ij} = c_{i0} + c_{0j} - c_{ij}, \qquad (11)$$

em que R_{ij} é a redução da distância (e consequentemente do custo) que resulta do atendimento aos nós i e j em uma rota, em oposição a atendê-los individualmente em duas rotas distintas. Uma vez que esse valor é determinado para todos os arcos possíveis, a redução é classificada em ordem decrescente e os arcos são adicionados, um por um, à rota parcialmente formada, garantindo que as restrições de tempo e capacidade sejam atendidas em cada iteração. Vale lembrar que devido à existência de janelas de tempo, a orientação das rotas também deve ser considerada, uma vez que a adição de um arco (i,j) pode ser aceita na rota, mas não necessariamente a de seu inverso (j,i).

A Figura 10 ilustra o CPR ou nó 0, junto com duas fazendas 1 e 2 que são conectadas por uma estrada, o arco $(1,2)$. A título de exemplo, são atribuídos os custos $c_{01}=c_{10}=10$, $c_{02}=c_{20}=20$ e $c_{12}=c_{21}=15$. Neste caso, a redução de custo alcançada é $R_{12}=R_{21}=c_{10}+c_{02}-c_{12}=10+20-15=15$.

Figura 10 – Exemplo no cálculo na redução de custos.



Fonte: O autor.

Segundo Toth et al. (2002), uma desvantagem do algoritmo original é que produz boas rotas no começo da sua execução, mas outras muito menos interesantes no final, incluindo algumas rotas circunferenciais. Para amenizar esse inconveniente, foi proposta a inclusão de un parâmetro μ no cálculo da redução de custos, chamado de parâmetro de forma. Quanto maior μ, mais importância é dada para as distâncias entre os nós a serem conectados. Diversos autores relatam que usualmente $\mu=0.4$ ou 1.0 conduz a boas soluções, levando em consideração o número de rotas e o comprimento total da solução.

$$R_{ij}=c_{i0}+c_{0j}-\mu c_{ij} \tag{12}$$

Partindo das distâncias entre nós, C_{ij}, o algoritmo de redução de custos pode ser descrito como segue (LARSON, 1981; CACETTA, 2013):

Etapa 1

Dado μ, calcule a redução de custos $R_{ij}=c_{i0}+c_{0j}-\mu c_{ij}$ para $i,j=1\ldots n$ e $i\neq j$.

Etapa 2

Classifique os fatores de redução R_{ij} e liste-os em ordem decrescente de magnitude. Isso cria a "lista de redução". Processe a lista de redução começando com a primeira entrada da lista $(\text{o maior } R_{ij})$.

Etapa 3 (enquanto ainda houver membros da lista a serem processados)

Para a redução R_{ij} em consideração, inclua a conexão (i,j) em uma rota se nenhuma restrição de rota, temporal ou de capacidade, for violada através da inclusão de (i,j). Os três casos a seguir precisam ser considerados:

a. Se nem i nem j já foram atribuídos a uma rota, então uma nova rota é iniciada incluindo i e j.

b. Se exatamente um dos dois pontos $(i \text{ ou } j)$ já foi incluído em uma rota existente e esse ponto não é interior a essa rota (um ponto é interior a uma rota se não for adjacente ao CPR na ordem de travessia de pontos), então a conexão (i,j) é adicionada a essa mesma rota.

c. Se i e j já tiverem sido incluídos em duas rotas diferentes e nenhum deles for interno à sua rota, as duas rotas serão fundidas conectando i e j. Se eles forem internos, a fusão não poderá ser feita.

Processe o próximo termo R_{ij} da lista de redução.

Etapa 4

A solução para o *VRPTW* consiste nas rotas criadas durante a ***Etapa 3***. (Todos os pontos que não foram atribuídos a uma rota durante a ***Etapa 3*** devem ser servidos por uma rota de veículo que começa no CPR, visita o ponto não atribuído e retorna para o CPR).

O APÊNDICE A contém a implementação computacional do algoritmo de redução de custos, junto com os processos de entrada e saída de dados, em linguagem C. A implementação foi desenvolvida em um computador Intel® Core™ i7-2630QM CPU @2.00 GHz x 8 com 5.7 GB de memória RAM e sistema operacional Debian GNU/Linux 10 (buster) de 64 bits. O compilador usado foi o gcc versão 8.3.0 (Debian 8.3.0-6) e deve ser chamado da seguinte maneira:

```
gcc savings.c -lm

./a.out <Dataset> <Número de clientes a considerar> 1.0
```

3.2.2 Heurística de inserção I1

Este tipo de heurística, também proposta por Solomon (1987) para solucionar o *VPRTW,* constrói as rotas sequencialmente. Para isso cada rota começa ou com o ponto de coleta mais distante ou com aquele cujo serviço termina antes, ou seja, cada rota começa com um cliente i que ainda não foi visitado, para o qual a distância C_{0i} (critério 1) é máxima ou o valor de b_i é mínimo (critério 2). Depois de iniciar uma rota, o método calcula c_1 e c_2 em cada iteração, dois valores que permitem selecionar o nó u a inserido entre os nós i e j, já presentes na rota em construção, conforme Equações (13) e (14)

Seja $(i_0, i_1, \ldots, i_m)$ a rota em construção, com $i_0 = i_m = 0$. Para cada nó não visitado, calcula-se dentro da rota em questão o melhor local de inserção possível, no qual

$$c_1(i(u), u, j(u)) = \min[c_1(i_{p-1}, u, i_p)], \ p = 1, \ldots, m. \tag{13}$$

Em seguida, é escolhido o melhor cliente a ser inserido, para o qual

$$c_2(i(u^*), u^*, j(u^*)) = \max[c_2(i(u), u, j(u))], \ u \text{ no roteado e viável}. \tag{14}$$

Assim, o nó u^* é adicionado à rota entre $i(u^*)$ e $j(u^*)$. Quando não forem encontrados mais nós com inserções possíveis, o método inicia uma nova rota, a menos que todos os pontos de coleta já tenham sido visitados. O método de inserção I1 calcula c_1 e c_2 tendo em conta:

$$c_{11}(i, u, j) = C_{iu} + C_{uj} - \mu C_{ij}, \ \mu \geq 0; \tag{15}$$

$$c_{12}(i, u, j) = b_{j_u} - b_j, \tag{16}$$

onde b_{j_u} é o novo tempo em que o serviço começa no nó j, dado que u está na rota;

$$c_1(i, u, j) = \alpha_1 c_{11}(i, u, j) + \alpha_2 c_{12}(i, u, j); \ \alpha_1 + \alpha_2 = 1; \ \alpha_1 \geq 0, \ \alpha_2 \geq 0; \tag{17}$$

$$c_2(i, u, j) = \lambda C_{0u} - c_1(i, u, j), \ \lambda \geq 0. \tag{18}$$

Este tipo de heurística de inserção tenta maximizar o benefício derivado de servir um nó na rota parcial em construção em oposição a atendê-lo em uma rota direta. Por exemplo, quando $\mu = \alpha_1 = \lambda = 1$ e $\alpha_2 = 0$, c_2 corresponde à distância poupada visitando o nó u na mesma rota com os nós i e j, em comparação com uma

rota direta. O melhor local para a possível inserção de um nó não visitado é aquele que minimiza a combinação ponderada da sua distância e tempo de inserção, ou seja, aquele que minimiza a medida da distância adicional e do tempo adicional, necessários para visitar o referido nó. Claramente, valores diferentes de μ e λ levam a critérios diferentes para selecionar um nó possível e seu melhor local de inserção.

O algoritmo de inserção I1 pode ser resumido da seguinte forma:

Etapa 1

Inicie uma nova rota com

(critério 1) o nó não roteado mais distante ou

(critério 2) o nó não roteado com o prazo mais próximo.

Etapa 2

Para cada nó não roteado e factível, ou seja, aqueles cuja inserção na rota não viole nenhuma restrição de rota, temporal ou de capacidade;

determine o seu melhor local de inserção possível na rota com (13), para isso calcule também (15), (16) e (17). Este local é aquele que minimiza o valor de c_1 e

selecione o melhor nó a ser inserido na rota, no local já determinado, ou seja, aquele que maximiza c_2. Para este propósito, use (14) e (18).

Etapa 3

Insira o nó selecionado no local já definido e volte à ***Etapa 2***. Quando não forem encontrados mais nós com inserções possíveis, inicie uma nova rota (retorne à ***Etapa 1***) a menos que todos os nós tenham sido roteados.

Etapa 4

A solução para o *VRPTW* consiste nas rotas criadas durante as etapas anteriores.

O APÊNDICE B contém a implementação computacional do algoritmo de inserção I1, junto com os processos de entrada e saída de dados, em linguagem C. A implementação foi desenvolvida em um computador Intel® CoreTM i7-7700HQ CPU @2.80 GHz x 8 com 12 GB de memória RAM e sistema operacional Ubuntu 20.04 WSL 4.19.128-microsoft-standard #1 SMP GNU/Linux de 64 bits. O compilador usado foi o gcc versão 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04) e deve ser chamado da seguinte maneira:

gcc insertion.c -lm

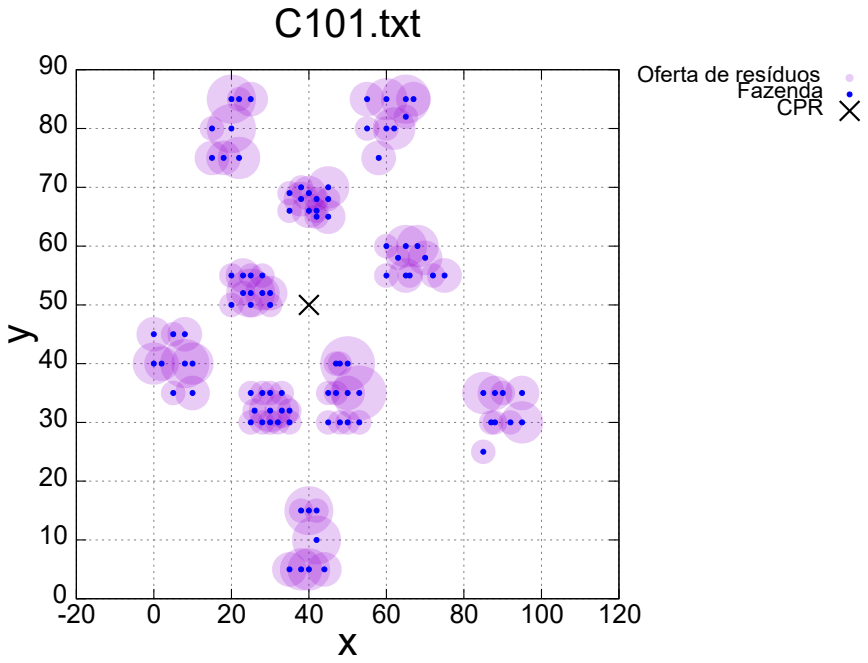
./a.out <Dataset> <Número de clientes a considerar> <Critério: 1 ou 2> <alpha_1> <alpha_2> <lambda> <mu>

3.2.3 Instâncias computacionais

Para avaliar o desempenho da metodologia proposta foram utilizados os conjuntos de dados gerados por Solomon (1987), os quais também são utilizados para *benchmarking* de algoritmos de roteamento (JIN et al. 2021). Esses conjuntos de dados levam em consideração fatores que podem afetar o comportamento das rotas e seu planejamento. Tais fatores incluem: informações geográficas, número de pontos de coleta atendidos por um veículo e características dos intervalos (janelas) de tempo nos quais os resíduos orgânicos ficam disponíveis: a porcentagem de fazendas com restrições de tempo, o instante no qual o caminhão pode iniciar o serviço na fazenda e a duração deste. Os conjuntos de dados correspondem a 56 problemas de *benchmark* ou instâncias, agrupados em seis categorias (R1, R2, C1, C2, RC1 e RC2), os quais apresentam entre oito e 100 pontos de coleta. Os conjuntos de dados R têm localizações de nós aleatórias, geradas uniformemente. Os conjuntos de dados C têm nós agrupados e janelas de tempo geradas com base em soluções conhecidas. Os conjuntos de dados RC apresentam uma mistura de nós localizados aleatoriamente e em grupos. Os conjuntos de dados do tipo 1 têm janelas de tempo estreitas e veículos de baixa capacidade. Os conjuntos de dados do tipo 2 têm amplas janelas de tempo e alta capacidade de carga para seus veículos. Portanto, as soluções de problemas do tipo 2 têm menos rotas e significativamente mais fazendas por rota. Estas diferentes categorias, representam diferentes situações reais com relação as localizações das fazendas, número de pontos de coletas, capacidade dos caminhões, período de tempo em que o resíduo estará disponível para coleta, etc.

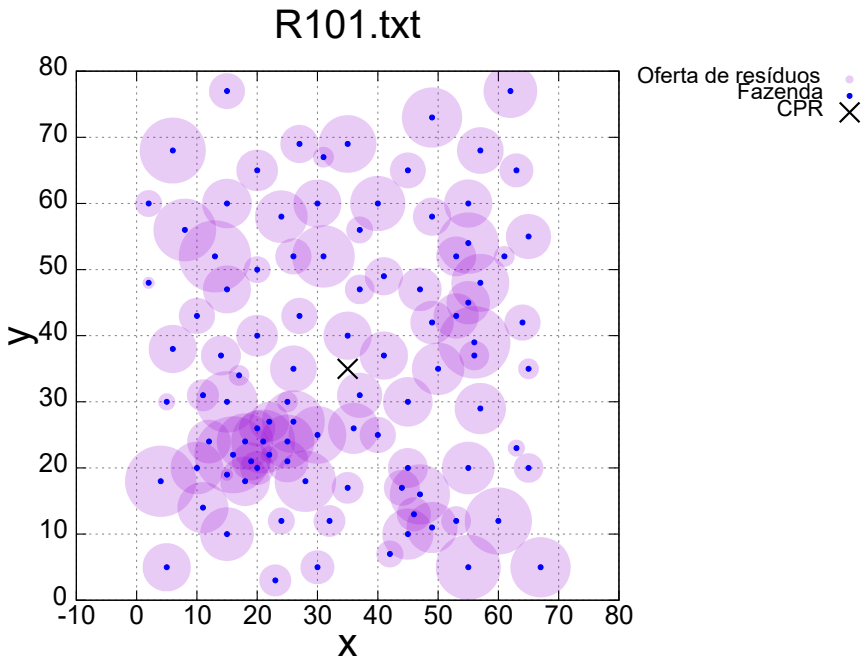
Essas diferenças podem ser observadas nas três instâncias apresentadas: Uma da categoria C, mostrada na Figura 11; uma da categoria R, mostrada na Figura 12, e uma da RC na Figura 13. As figuras apresentam a localização geográfica, representada por um ponto e a quantidade de resíduo disponível da fazenda, representada por um círculo de tamanho proporcional a esta. Adicionalmente, o CPR é mostrado com um X. Note que a disponibilidade de resíduo associada a este é zero, pois não há resíduo a ser coletado.

Figura 11 – Distribuição espacial dos pontos de coleta e disponibilidade de resíduo Instância C101. Os pontos para coleta de resíduos estão agrupados (em nichos).



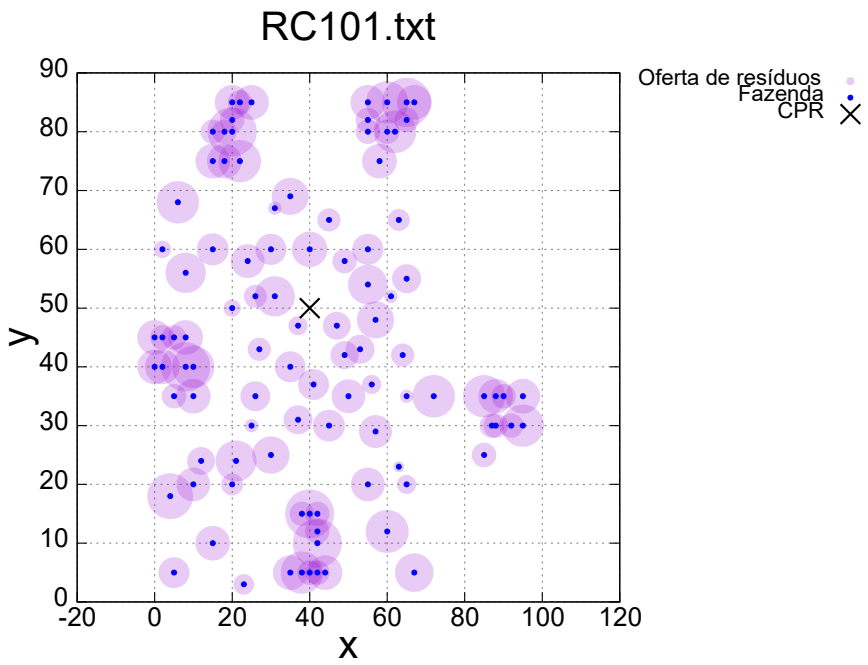
Fonte: O autor.

Figura 12 – Distribuição espacial dos pontos de coleta e disponibilidade de resíduo (Instância R101). Os pontos para coleta de resíduos estão espalhados.



Fonte: O autor.

Figura 13 – Distribuição espacial dos pontos de coleta e disponibilidade de resíduo (Instância RC101). Os pontos para coleta de resíduos estão alguns próximos uns dos outros e outros espalhados.



Fonte: O autor.

4 RESULTADOS E DISCUSSÃO

Os algoritmos de redução de custos e inserção I1 foram aplicados em 56 instâncias agrupadas nas categorias C1, C2, R1, R2, RC1 e RC2. Todas as execuções foram realizadas considerando 100 pontos de coleta, isto é, a versão completa das instâncias.

4.1 Heurística de redução de custos

A Tabela 3 apresenta os resultados obtidos para o roteamento de duas instâncias (R101 e R201) idênticas em termos de localização geográfica, mas com diferenças substanciais na capacidade dos caminhões para a coleta dos resíduos e nos intervalos de tempo que estes estarão disponíveis nas fazendas. Conforme mostra a Tabela 3, o algoritmo de redução de custos definiu 30 rotas para a instância R101 e 16 rotas para a instância R201, isto é devido as diferenças entre a capacidade dos caminhões e as janelas de tempos apresentadas para cada instância. O parâmetro de forma nesta execução foi $\mu = 1.0$. Como pode ser visto, o planejamento das rotas de coleta foi realizado com sucesso em todas as fazendas.

Tabela 3 – Roteamento obtido para as instâncias R101 e R201, usando o algoritmo de redução de custos.

	R101 – 30 rotas Capacidade dos caminhões: 200 (massa)		R201 – 16 rotas Capacidade dos caminhões: 1000 (massa)	
Índice de rota	**Resíduo recolhido (massa)**	**Pontos de coleta visitados**	**Resíduo recolhido (massa)**	**Pontos de coleta visitados**
1	35	0 65 71 0	142	0 59 5 45 47 63 65 71 0
2	49	0 63 64 49 0	150	0 92 72 14 83 36 62 19 11 64 49 0
3	32	0 36 47 0	105	0 42 27 95 15 39 67 0
4	56	0 39 67 0	163	0 28 31 33 29 79 81 51 9 78 34 35 0
5	45	0 33 29 78 34 35 0	143	0 69 12 2 98 61 16 44 38 86 0
6	54	0 14 44 38 0	91	0 75 52 88 30 20 66 0
7	57	0 62 11 90 32 0	97	0 23 82 99 18 7 90 32 0
8	62	0 72 23 55 25 0	84	0 76 21 40 73 43 56 54 24 55 25 0
9	88	0 27 30 51 9 66 0	61	0 6 85 84 8 46 0
10	66	0 92 98 16 86 0	70	0 53 87 57 41 22 0
11	48	0 59 42 15 43 0	76	0 94 96 97 37 91 100 0
12	17	0 45 46 0	68	0 50 3 68 80 0
13	31	0 2 75 56 0	38	0 26 10 70 0
14	95	0 5 82 19 48 0	60	0 1 13 4 74 0
15	33	0 83 61 84 17 0	78	0 89 48 60 17 93 0
16	66	0 95 21 73 22 74 0	32	0 77 58 0
17	77	0 52 99 85 91 100 0		
18	83	0 28 12 76 81 20 0		
19	42	0 69 79 3 0		
20	53	0 31 41 54 24 0		
21	64	0 88 50 68 80 0		
22	26	0 7 10 70 0		
23	56	0 87 37 93 0		
24	12	0 8 60 0		
25	28	0 40 57 97 0		

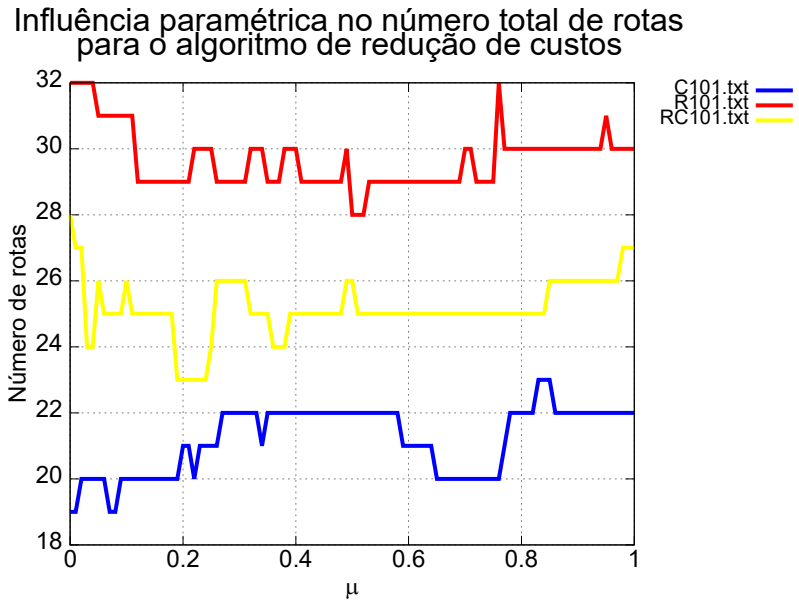
26	38	0 94 96 0
27	50	0 53 26 4 0
28	36	0 18 1 77 0
29	26	0 6 13 0
30	33	0 89 58 0

Fonte: O autor.

4.1.1 Análise paramétrica

Conforme descrito anteriormente, o parâmetro μ introduzido para melhorar o algoritmo de redução de custos, apresenta uma alta influência no número de rotas de coleta e seus comprimentos. Esse comportamento pode ser observado nas Figuras 14 e 15 para o número de rotas de coleta e distância total, respectivamente.

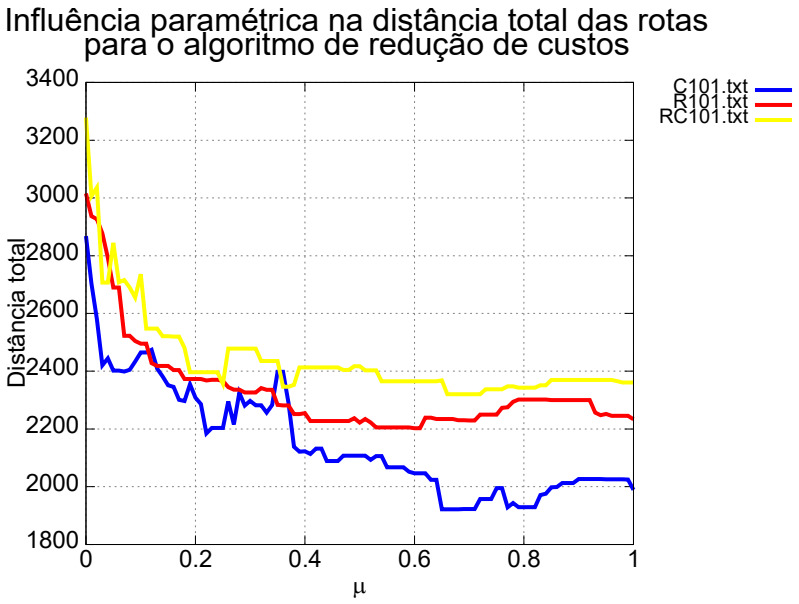
Figura 14 – Redução de custos: Influência paramétrica no número de rotas.



Fonte: O autor.

Pode-se observar na Figura 14 que as diferentes instâncias apresentaram comportamentos distintos para o número de rotas, dependendo do parâmetro μ. No caso do valor de μ próximo de 0.0, a instância C101 apresenta melhores resultados, enquanto que para o valor de μ próximo de 0.2, a instância RC101 apresenta melhores resultados e para o valor de μ nas proximidades de 0.5, obtém-se valores reduzidos para o número de rotas correspondentes à instância R101.

Figura 15 – Redução de custos: Influência paramétrica na distância total.



Fonte: O autor.

Os resultados apresentados na Figura 15 evidenciam que para μ próximo de 0.7, as três instâncias tratadas obtiveram um mínimo valor para a distância total percorrida. Embora diferentes valores de μ tenham proporcionado um número mínimo de rotas, neste trabalho é priorizada a otimização da distância total percorrida, para a qual o valor de μ nas proximidades de 0.7 é o mais adequado para as instâncias estudadas.

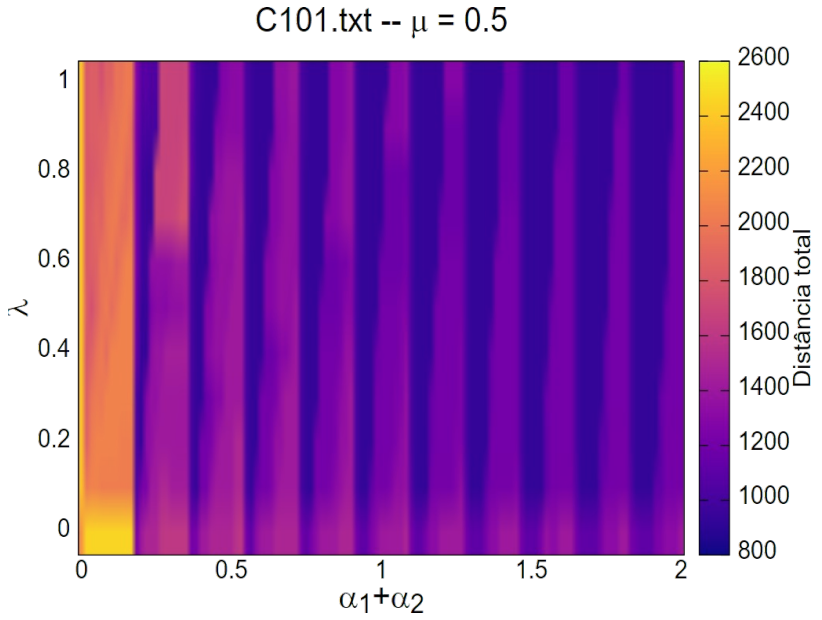
4.2 Heurística de inserção I1

Similarmente para o algoritmo de inserção I1, os parâmetros de entrada têm uma alta influência nos resultados, tanto no custo total quanto no número de rotas. Isto pode ser visto nas Figuras de 16 a 21, as quais evidenciam as diferenças obtidas na distância total percorrida e número de rotas para diferentes instâncias, utilizando dois valores diferentes de μ, enquanto valores de α_1, α_2 e λ mudam.

Instância C101

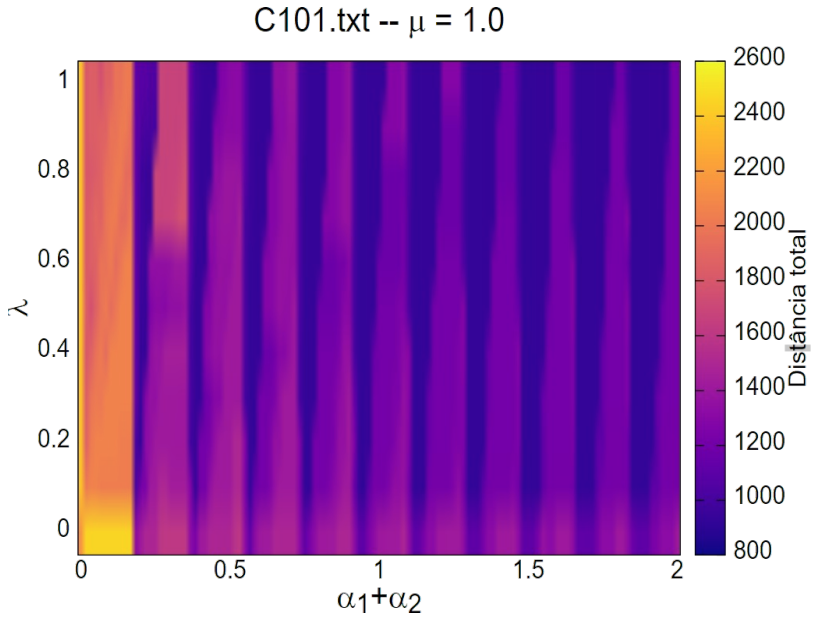
Foi analisada a influência dos parâmetros λ, α_1 e α_2 na distância total percorrida. Esses resultados estão apresentados nas Figuras 16 e 17 para $\mu=0.5$ e $\mu=1.0$, respectivamente. Estes valores de μ foram usados por serem os mais utilizados na literatura.

Figura 16 – Inserção I1: Influência paramétrica na distância total. C101: $\mu = 0.5$



Fonte: O autor.

Figura 17 – Inserção I1: Influência paramétrica na distância total. C101: $\mu = 1.0$



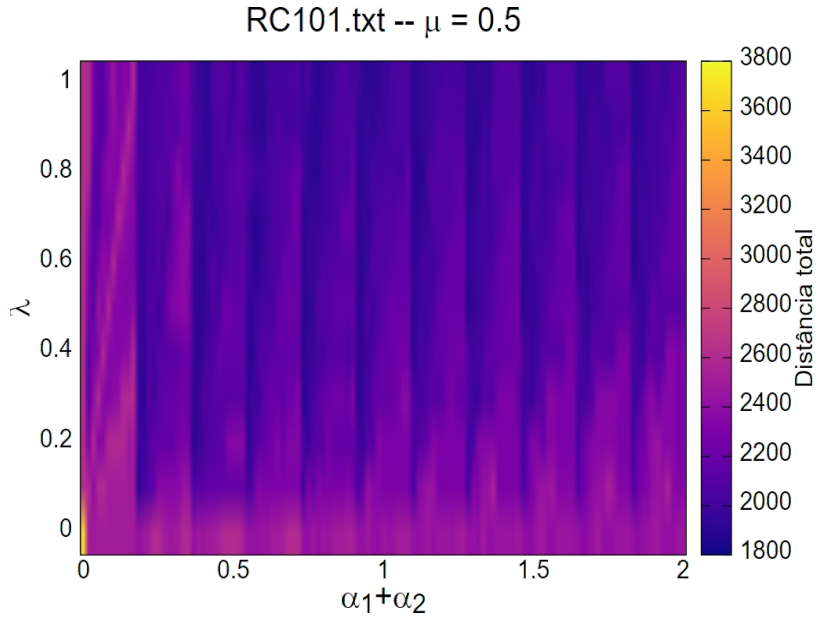
Fonte: O autor.

Para a instância C101, observa-se nas Figuras 16 e 17 que os melhores valores para α_1 e α_2, descritos por $(\alpha_1+\alpha_2)$, estão apresentados nas faixas mais escuras. A medida que esta soma aumenta, melhores reduções na distância total podem ser alcançadas quando aumenta o valor de λ.

Instância RC101

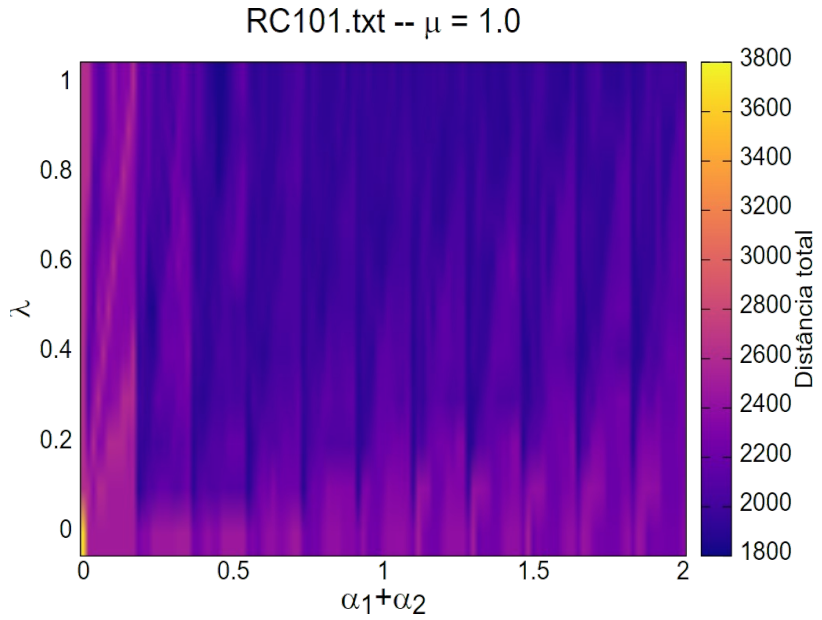
As Figuras 18 e 19 apresentam uma análise dos parâmetros λ, α_1 e α_2, visando estudar a influência destes na distância total percorrida na instância RC101, para os valores de $\mu=0.5$ e $\mu=1.0$, respectivamente.

Figura 18 – Inserção I1: Influência paramétrica na distância total. RC101: $\mu=0.5$



Fonte: O autor.

Figura 19 – Inserção I1: Influência paramétrica na distância total. RC101: $\mu=1.0$



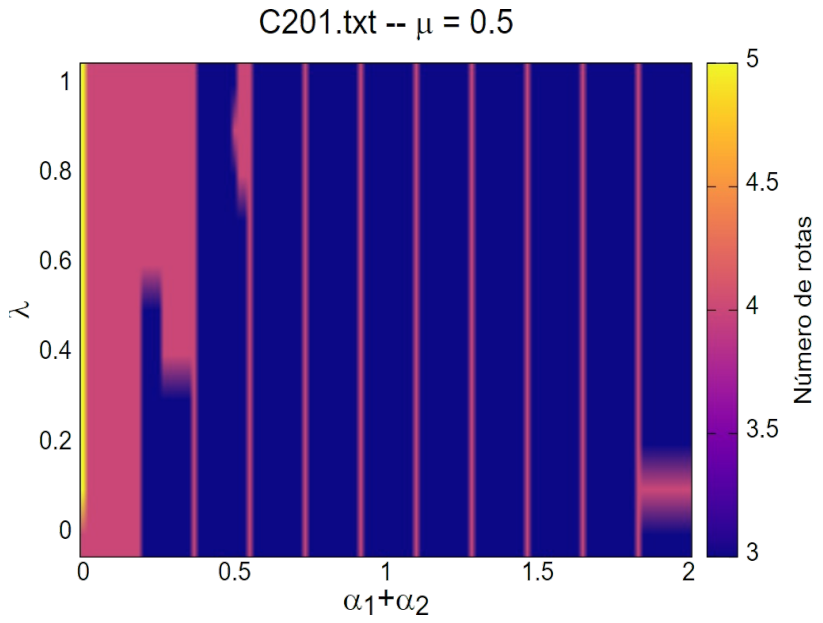
Fonte: O autor.

É importante notar que, para o caso da instâcia RC101 o padrão de listras é menos evidente do que na instância anterior. Pode ser observado também que o parâmetro λ, nesse caso, tem uma maior influência do que os parâmetros α_1 e α_2, sendo que para maiores valores de λ, obtém-se menores valores para a distância total, quando a soma $\alpha_1+\alpha_2$ é maior do que 0.4.

Instância C201

As Figuras 20 e 21 apresentam a influência da variação dos parâmetros λ, α_1 e α_2 no número de rotas, para a instância C201, utilizando respectivamente $\mu=0.5$ e $\mu=1.0$.

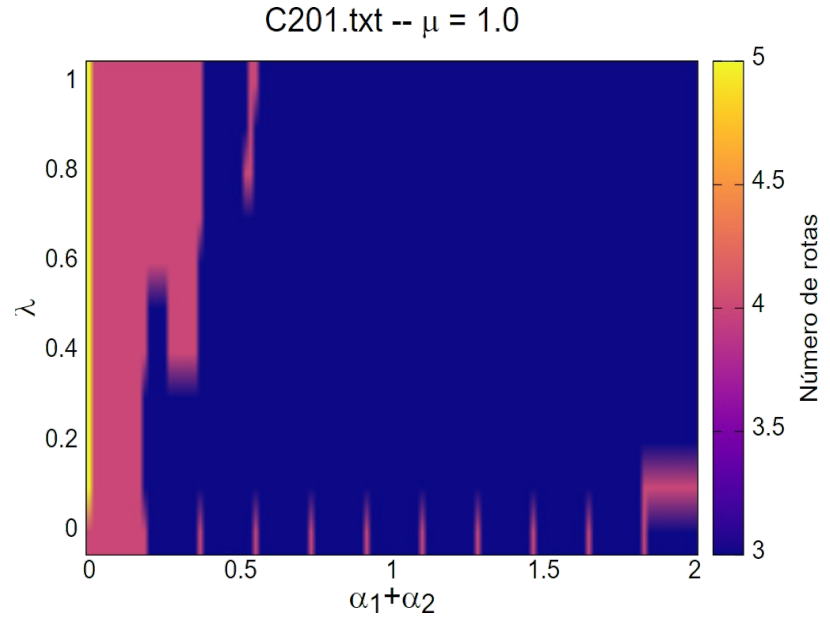
Figura 20 – Inserção I1: Influência paramétrica no número de rotas. C201: $\mu=0.5$



Fonte: O autor.

Observa-se que na região central da Figura 20 há uma padronização de listras, indicando que os valores de α_1 e α_2 têm maior importância do que os valores de λ para esta instância, sendo que os melhores valores para o número de rotas correspondem aos valores de $\alpha_1+\alpha_2$ representados pelas faixas escuras, uniformes e centrais da Figura 20.

Figura 21 – Inserção I1: Influência paramétrica nnúmero de rotas. C201: $\mu = 1.0$



Fonte: O autor.

A Figura 21 evidencia que essas faixas podem quase desaparecer conforme aumenta o valor de μ, nesse caso a região de resultados ótimos é evidentemente maior, dando assim mais flexibilidade na seleção dos outros parâmetros, mostrando, neste caso, que todos os parâmetros devem assumir maiores valores para se obter menor número de rotas.

4.2.1 Roteamento

A Tabela 4 mostra as rotas obtidas com o algoritmo de inserção I1. Foram obtidas 21 rotas para a instância R101 e 5 rotas para a instância R201. Neste caso, os parâmetros usados foram $\mu = \alpha_1 = \lambda = 1$ e $\alpha_2 = 0$. Nota-se uma diminuição significativa no número de rotas planejadas, em comparação com o algoritmo de redução de custos, o que se reflete na quantidade de caminhões a serem utilizados durante a coleta.

Tabela 4 – Roteamento obtido para as instâncias R101 e R201, usando o algoritmo de inserção I1.

Índice de rota	R101 – 21 rotas Capacidade dos caminhões: 200 (massa) Resíduo recolhido (massa)	Pontos de coleta visitados	R201 – 5 rotas Capacidade dos caminhões: 1000 (massa) Resíduo recolhido (massa)	Pontos de coleta visitados
1	87	0 65 71 9 34 35 77 0	492	0 92 59 5 27 28 33 65 69 71 29 12 76 78 9 81 51 79 6 3 50 34 66 20 35 24 68 4 74 80 77 70 93 100 58 0
2	103	0 63 64 49 48 58 0	394	0 72 42 14 95 83 45 82 36 47 19 62 11 64 7 88 18 8 90 49 10 46 96 1 32 48 17 91 60 89 25 0
3	112	0 39 23 67 55 4 25 0	329	0 63 2 15 39 21 67 23 75 73 40 99 85 87 57 22 41 56 55 54 26 13 0
4	79	0 14 44 38 43 91 100 0	208	0 31 52 98 38 44 16 61 84 86 94 43 37 97 0
5	96	0 36 47 19 90 10 32 70 0	35	0 30 53 0
6	83	0 33 30 51 66 1 80 0		
7	94	0 59 83 82 7 8 46 17 93 0		
8	117	0 92 42 98 16 86 37 13 89 0		
9	40	0 62 11 20 0		
10	80	0 28 29 78 3 68 24 0		
11	76	0 95 15 87 57 97 60 0		
12	84	0 72 75 73 22 56 74 0		
13	47	0 45 61 84 96 0		
14	40	0 2 21 41 26 0		
15	87	0 27 12 76 81 50 0		
16	47	0 69 79 54 0		
17	76	0 5 99 85 0		
18	39	0 31 88 6 0		
19	21	0 52 18 0		
20	36	0 40 94 0		
21	14	0 53 0		

Fonte: O autor.

Foram realizados também os roteamentos para as outras 54 instâncias. A Tabela 5 apresenta um resumo dos resultados obtidos para o número médio de rotas e custo (distância total) destas 56 instâncias, agrupados por categoria e diferenciados para cada um dos algoritmos implementados. Adicionalmente, os valores obtidos neste trabalho são comparados com os resultados obtidos por Solomon (1987), usando o método de inserção I1.

Tabela 5 – Número de rotas e custo total obtido para as heurísticas implementadas.

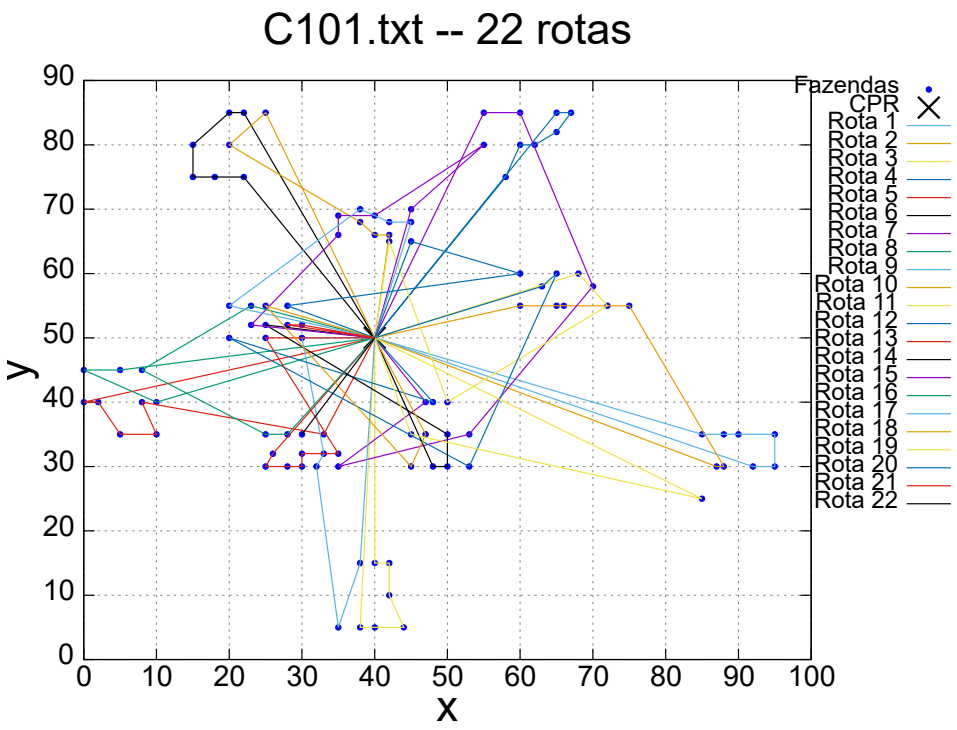
	Instâncias	Redução de custos		Inserção I1		Inserção I1 Solomon (1897)	
		$\bar{n}$	$\bar{C}$	$\bar{n}$	$\bar{C}$	$\bar{n}$	$\bar{C}$
Valores absolutos	C1	18.4	1762.4	10.1	941.2	10	951.9
	C2	13.6	1893.2	3.1	674.1	3.1	692.7
	R1	20.1	1750.7	13.9	1380.0	13.6	1436.7
	R2	10.6	1791.6	3.5	1258.9	3.3	1402.4
	RC1	19.6	2004.4	13.9	1578.6	13.5	1596.5
	RC2	11.1	2198.6	4.0	1541.3	3.9	1682.1
Desvio (%)	C1	84.0	85.1	1.0	-1.1	-	-
	C2	338.7	173.3	0.0	-2.7	-	-
	R1	47.8	21.9	2.2	-3.9	-	-
	R2	221.2	27.8	6.1	-10.2	-	-
	RC1	45.2	25.5	3.0	-1.1	-	-
	RC2	184.6	30.7	2.6	-8.4	-	-

$\bar{n}$: Número médio de rotas planejadas $\bar{C}$: Custo médio total (distância)

Fonte: O autor.

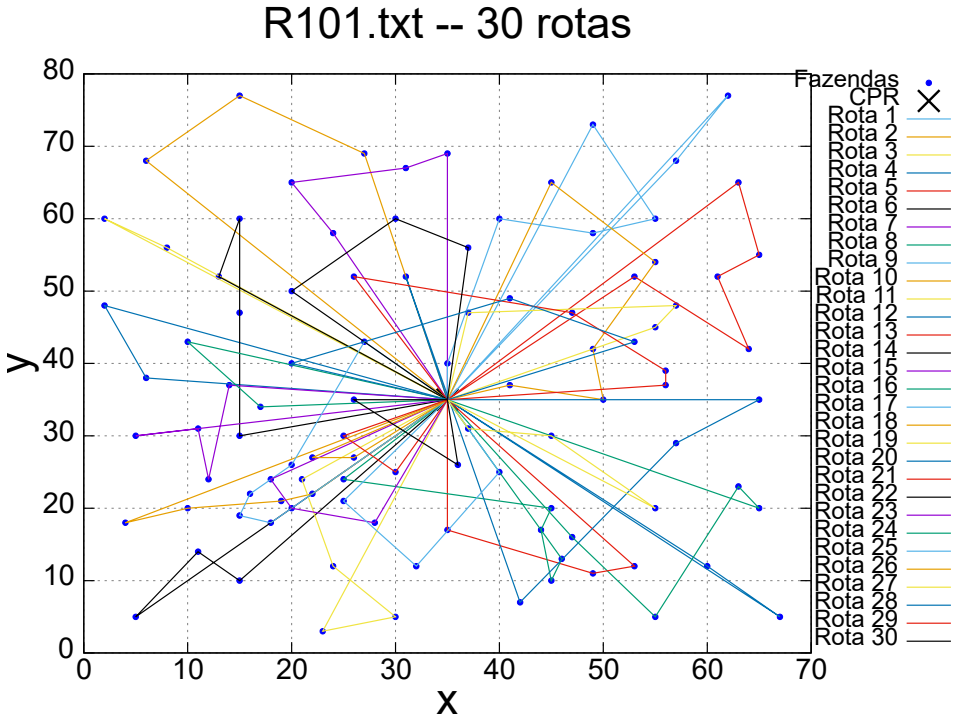
Pode ser observado na Tabela 5 que o número médio de rotas obtidas para dados do tipo C é significativamente menor do que para os dados do tipo R, da mesma forma os conjuntos de dados 1 requerem um maior número de rotas do que os conjuntos de dados do tipo 2, uma vez que as janelas de tempo estreitas nas quais as fazendas podem ser atendidas não permitem que o mesmo caminhão se mova de um local para outro sem violar as restrições de tempo. Também deve-se observar que houve uma redução na distância total é alcançada para todos os grupos de dados, em comparação com os valores de Solomon (1987), sendo o mais evidente aquele correspondente ao grupo R2, em que houve uma redução média de 10.1%. Isso se deve à busca sistemática pelo melhor valor possível, uma vez que a execução se repete enquanto os parâmetros são variados. Os resultados alcançados evidenciam as vantagens dos métodos heurísticos, pois permitem manipular um grande número de dados com facilidade e obter uma solução rapidamente. Observa-se também um melhor desempenho do algoritmo de inserção I1 devido aos seus critérios de avaliação, tanto para a posição na rota quanto para o ponto de coleta a ser inserido nela. Este comportamento está ilustrado nas Figuras 22, 23, 24 e 25, nas quais pode-se ver a quantidade de rotas geradas pelo algoritmo de redução de custo e seu mapeamento nos pontos de coleta, para as instâncias C101, R101, R201 e RC101, respectivamente.

Figura 22 – Roteamento para a instância C101, heurística de redução de custos.



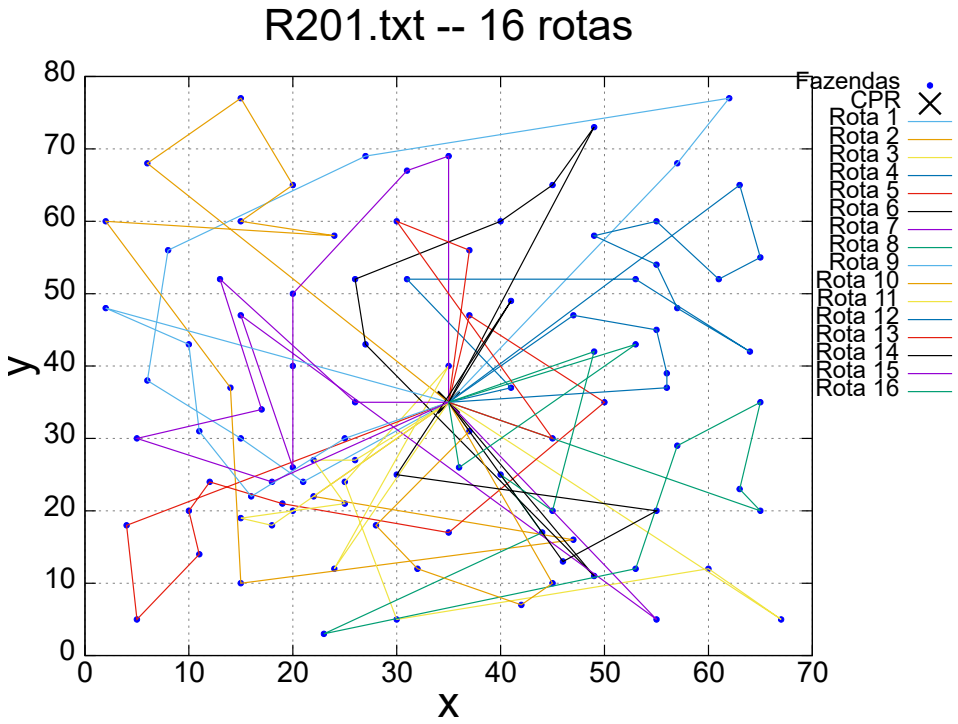
Fonte: O autor.

Figura 23 – Roteamento para a instância R101, heurística de redução de custos.



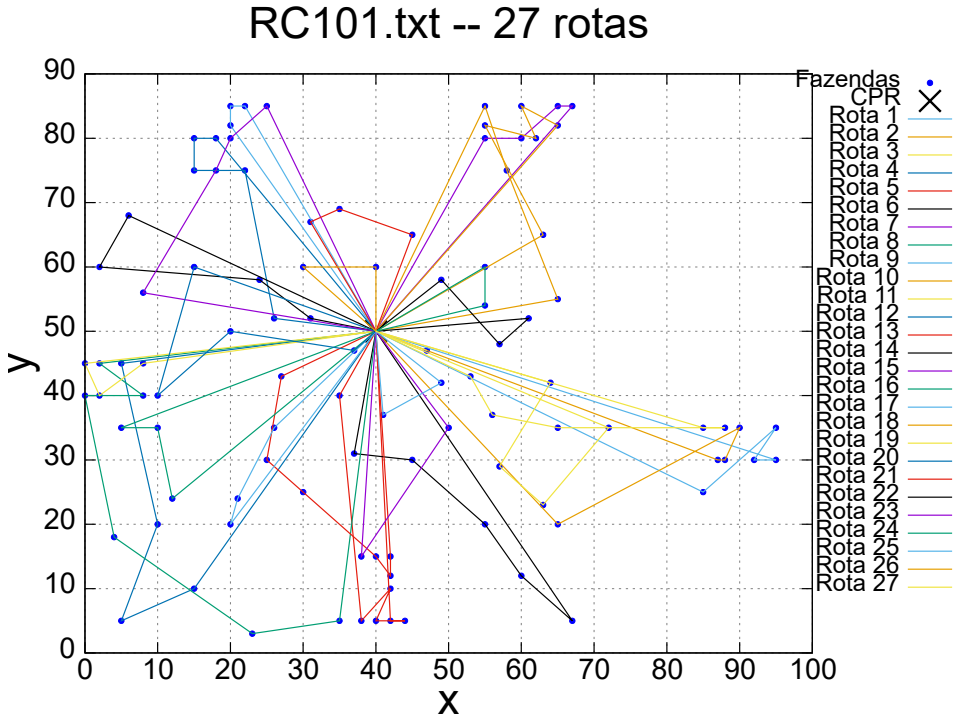
Fonte: O autor.

Figura 24 – Roteamento para a instância R201, heurística de redução de custos.



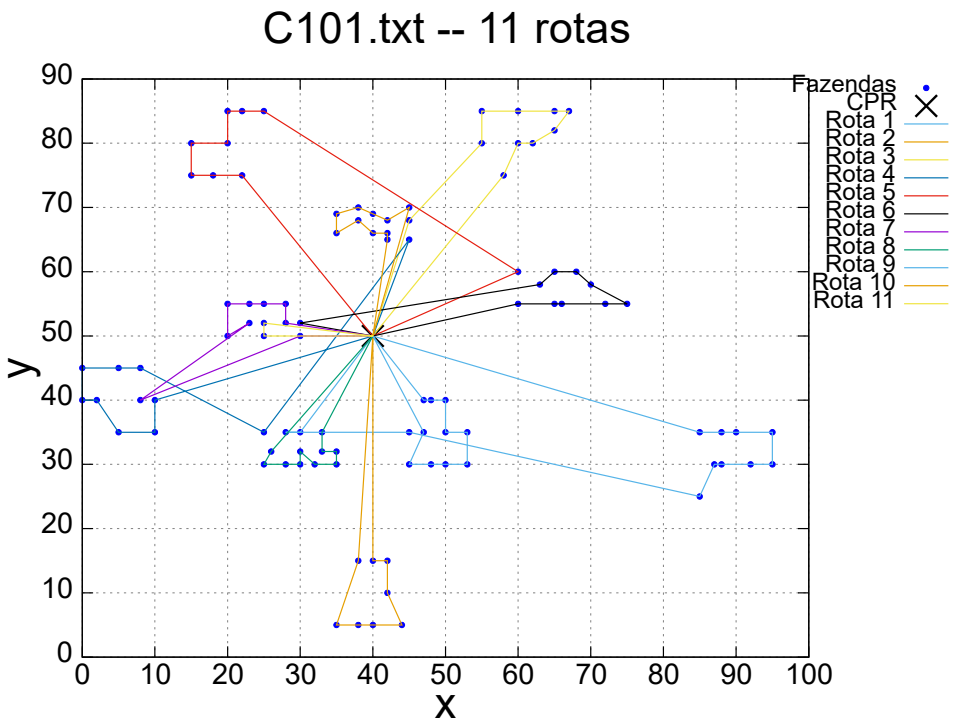
Fonte: O autor.

Figura 25 – Roteamento para a instância RC101, heurística de redução de custos.



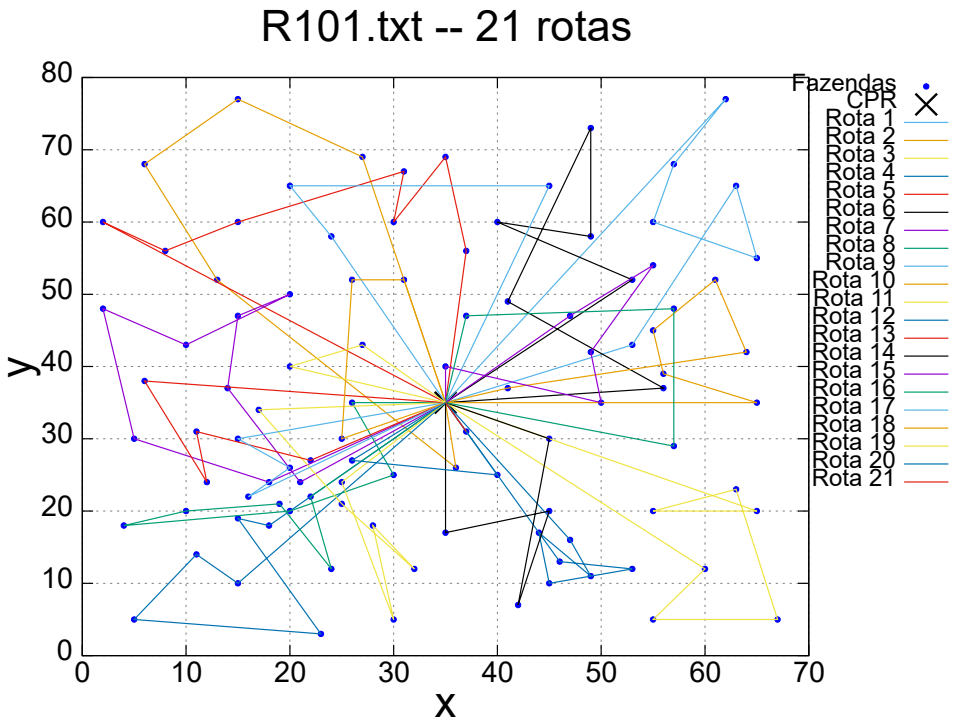
Fonte: O autor.

Figura 26 – Roteamento para a instância C101, heurística de inserção I1.



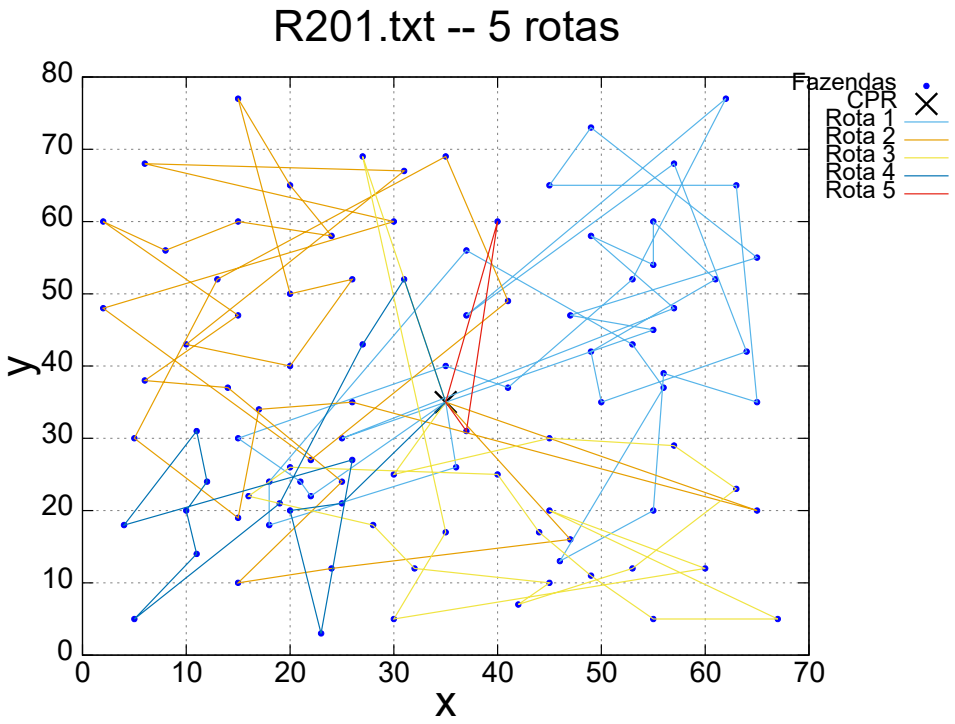
Fonte: O autor.

Figura 27 – Roteamento para a instância R101, heurística de inserção I1.



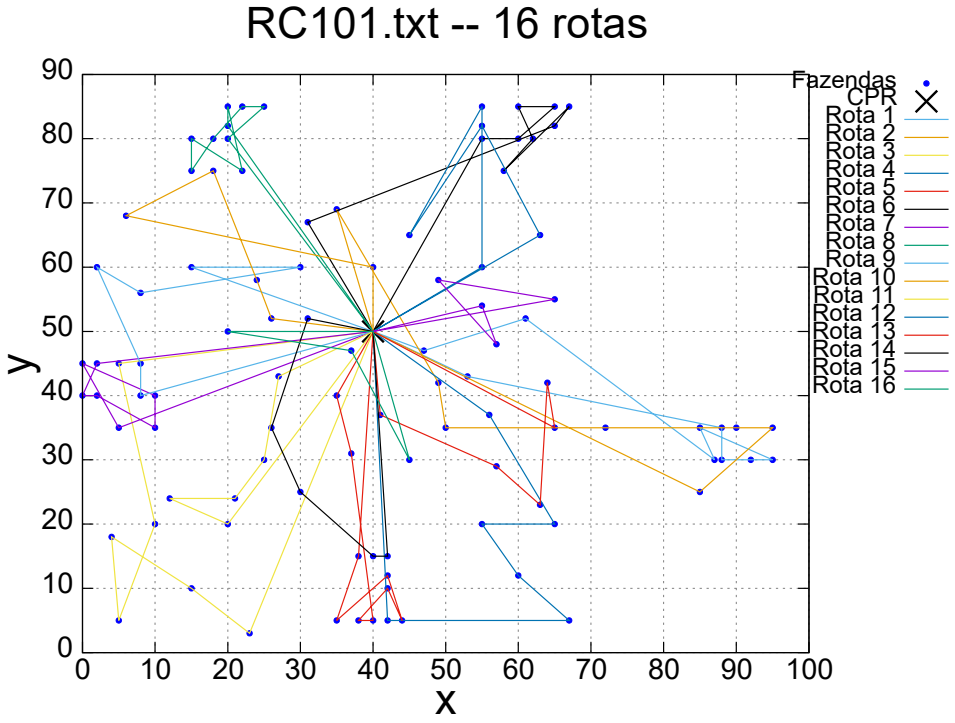
Fonte: O autor.

Figura 28 – Roteamento para a instância R201, heurística de inserção I1.



Fonte: O autor.

Figura 29 – Roteamento para a instância RC101, heurística de inserção I1.



Fonte: O autor.

A Tabela 6 mostra uma comparação dos tempos usados na obtenção da solução para cada conjunto de dados.

Tabela 6 – Tempos computacionais médios por instância para o cálculo e determinação da melhor solução disponível.

	Redução de custos	Redução de custos Solomon (1987)	Inserção I1	Inserção I1 Solomon (1987)
Tempo computacional total (s)	26.409	-	260.720	-
Tempo computacional de uma execução (s)	0.264	2.4	0.100	38.512

Fonte: O autor.

No método de redução de custo 100, execuções foram realizadas (variando μ a cada 0.01) e para o método de inserção I1 são realizadas 2.592 execuções, combinando os dois critérios para inserir o primer nó na rota (mais longe ou que deva ser atendido mais cedo) e os parâmetros α_1, α_2, λ e μ variando a cada 0.2. Nota-se uma redução considerável nos tempos computacionais por execução, em comparação com aqueles obtidos por Solomon (1987). Embora o método de redução de custos seja mais simples, ele possui um tempo de execução maior, ao contrário dos resultados apresentados por Solomon (1987). Isso se deve à implementação da versão sequencial do algoritmo de redução de custos.

Tanto o algoritmo de redução de custo quanto o algoritmo de inserção I1 foram capazes de determinar rotas para o problema logístico *VRPTW*, obedecendo as janelas de tempo, atendendo as demandas de resíduos a serem coletados e respeitando a capacidade de carga dos caminhões para as diferentes características das instâncias.

5 CONCLUSÕES

A metodologia proposta provou ser facilmente adaptável a problemas de logística e logística reversa, como neste caso. Além disso, a sua aplicação no setor rural e na gestão de resíduos no geral apresenta um amplo potencial, que não se limita a este domínio, mas sim à gestão de culturas, distribuição de produtos, entre outros.

Embora as melhorias alcançadas correspondam apenas ao processo de coleta de resíduos orgânicos para biodigestores, deve-se lembrar que a eficiência de um sistema é só o produto das eficiências de seus componentes, portanto qualquer melhoria individual, neste caso na coleta de resíduos, contribui para o processo de produção de energia e os trabalhos realizados nas fazendas que dela dependem.

As heurísticas aplicadas, apresentam grande potencial para resolução do problema de coleta de resíduos, devido à flexibilidade com que tratam grandes conjuntos de dados e à velocidade com que calculam as soluções, sendo muito adequada à gestão de resíduos urbanos.

É sabido que o problema em questão apresenta grande complexidade computacional e que os métodos exatos são inviáveis para o número de fazendas (pontos de coleta) abordados nas instâncias tratadas e nos casos reais, portanto o algoritmo de redução de custos e o algoritmo de inserção I1 podem ser usados como uma ferramenta de auxílio a tomada de decisões em empresas de produção de biogás a partir de resíduos orgânicos.

Uma perspectiva futura é aprofundar a pesquisa nos métodos heurísticos, buscando maiores informações da influência dos valores dos parâmetros, assim como investigar outras técnicas que permitam comparar o desempenho dos algoritmos implementados.

REFERÊNCIAS

AGHA, SALAH R. Optimizing routing of municipal solid waste collection vehicles in Deir el-Balah--Gaza strip. **Optimizing Routing Of Municipal Solid Waste Collection Vehicles in Deir El-Balah--Gaza Strip**, v. 14, n. 2, 2006.

AGNU. Transforming our world: the 2030 Agenda for Sustainable Development. **Division for Sustainable Development Goals: New York, NY, USA**, v. , n. , .

AGUILAR, F. E BOTERO, R.. Los beneficios económicos totales de la producción de biogas utilizando un biodigestor de polietileno de bajo costo. Economic benefits of biogas production using a low-cost polyethylene biodigester. **Tierra Tropical: Sostenibilidad, Ambiente y Sociedad**, v. 2, n. 1, 2006.

ARCHETTI, C. FEILLET, D. GENDREAU, M. E SPERANZA, M. Complexity of the VRP and SDVRP. **Transportation Research Part C: Emerging Technologies**, v. 19, n. 5, 2009.

ARINGHIERI, ROBERTO AND BRUGLIERI, MAURIZIO AND MALUCELLI, FEDERICO AND NONATO, MADDALENA. An asymmetric vehicle routing problem arising in the collection and disposal of special waste. **Electronic notes in discrete mathematics**, v. Electronic notes in discrete mathematics, n. , 2004.

AWAD, A. R. AND ABOUL-ELA, M.T. AND ABU-HASSAN, R.. Development of a simplified procedure for routering solid waste collection. , v. 55, n. 4, 2001.

BAZARAA, M. S., JARVIS, J. J., E SHERALI, H. D, **Linear programming and network flows**. N.J: John Wiley & Sons, 2009. 748 p. ISBN 978-0-470-46272-0.

CACCETTA, L. ALAMEEN, M. E ABDUL-NIBY, M.. An improved clarke and wright algorithm to solve the capacitated vehicle routing problem. **Engineering, Technology & Applied Science Research**, v. 3, n. 2, 2013.

CANTRELL, K.B. DUCEY, T. RO, K.S. E HUNT, P.G. Livestock waste-to-bioenergy generation opportunities. **Bioresource technology**, v. 99, n. 17, 2008.

CAO, B. Solving vehicle routing problems using an enhanced clarke-wright algorithm: a case study. **International Conference on Computational Logistics**, v. , n. , 2012.

CHAUDHARI, P. MISHRA, I. E CHAND, S. Decolourization and removal of chemical oxygen demand (COD) with energy recovery: treatment of biodigester effluent of a molasses-based alcohol distillery using inorganic coagulants. **olloids and Surfaces A: Physicochemical and Engineering Aspects**, v. 296, n. 1-3, 2007.

CHRISTOPHER, MARTIN, **Logistics & Supply Chain Management**. Harlow, UK: Pearson, 2011. 276 p. ISBN 978-0-273-73112-2.

CLARK, ROBERT M. AND GILLEAN, JAMES I.. Analysis of solid waste management operations in Cleveland, Ohio: A case study. **Interfaces**, v. 6, n. 1-2, 1975.

CLARKE, G. E WRIGHT, J. Scheduling of vehicles from a central depot to a number of delivery points. **Operations research**, v. 12, n. 4, 1964.

CLAY MATHEMATICS INSTITUTE, **Millennium Problems**, 2021, disponível em <https://www.claymath.org/millennium-problems>. Acesso em: 01, mar. 2021, 15:01:20.

CLEMENS, HARRY E BAILIS, ROB E NYAMBANE, ANNE E NDUNG'U, VICTORIA. Africa Biogas Partnership Program: A review of clean cooking implementation through market development in East Africa. **Energy for Sustainable Development**, v. 46, n. , 2018.

CLIMATE TECHNOLOGY & NETWORK CENTRE, **Biodigester**, 2016, disponível em <https://www.ctc-n.org/technologies/biodigester>. Acesso em: 17, ago. 2020, 14:31:45.

COOK, STEPHEN. The P versus NP problem. **The millennium prize problems**, v. , n. , 2006.

CORDEAU, J. DESAULNIERS, G. DESROSIERS, J. SOLOMON, M. E SOUMIS, F, **The VRP with Time Windows**. Montreal, Quebec: SIAM, 2002. 157-193 p. ISBN 0-89871-498-2.

CORDEAU, J., LAPORTE, G., SAVELSBERGH, M. W.P, AND VIGO, D.. Chapter 6: Vehicle Routing. **Handbooks in operations research and management science: transportation**, v. 14, n. , 2007.

DAS, S. AND BHATTACHARYYA, B. K.. Optimization of municipal solid waste collection and transportation routes. **Waste Management**, v. 43, n. , 2015.

DELL'AMICO, M., RIGHINI, G. AND SALANI, M.. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. **Transportation science**, v. 40, n. 2, 2006.

DESROCHERS, M. LENSTRA, J. SAVELSBERGH, M. E SOUMIS, F, **Vehicle routing with time windows: optimization and approximation**, 1987.

DORNELAS, K. SCHNEIDER, R. E DO AMARAL, A. Biogas from poultry waste—production and energy potential. **Environmental monitoring and assessment**, v. 189, n. 8, 2017.

EDWARDS, D. E HAMSON, M, **Guide to mathematical modelling**. : Macmillan International Higher Education, 2016. 277 p. ISBN 978-1-349-10042-2.

FLEISHMANN, MORITZ, **Quantitative models for reverse logistics**. Berlim: Springer-Verlag Berlin Heidelberg, 2001. 180 p. ISBN 978-3-540-41711-8.

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

GYAMFI, MATHIAS. Sequential Ordering Of Routes for Trucks for Efficient Garbage Collection: Case Study of Sekondi--Takoradi Metropolitan Assembly (STMA). , v. , n. , 2012.

HAN, HUI AND PONCE CUETO, EVA. Waste collection vehicle routing problem: literature review. **Promet-Traffic\Transportation**, v. 27, n. 4, 2015.

JIN, Y. GE, X. AND ZHANG, L. E REN, J.. A two-stage algorithm for bi-objective logistics model of cash-in-transit vehicle routing problems with economic and environmental optimization based on real-time traffic data. **Journal of Industrial Information Integration**, v. , n. , 2021.

JOHANSSON, OLA M.. The effect of dynamic scheduling and routing in a solid waste management system. **Waste management**, v. 26, n. 8, .

KAPUR, J, **Mathematical modelling**. New Delhi: New Age International, 2015. p. ISBN 978-81-224-3807-9.

LANSING, S; BOTERO, R; MARTIN, J. Waste treatment and biogas quality in small-scale agricultural digesters. **Bioresource technology**, v. 99, n. 13, 2008.

LANSING, S. ET AL. Wastewater transformations and fertilizer value when co-digesting differing ratios of swine manure and used cooking grease in low-cost digesters. **Biomass and bioenergy**, v. 34, n. 12, 2010.

LARSON, R. E ODONI, A, **Urban Operations Research**. N.J: Prentice-Hall, 1981. 573 p. ISBN 0139394478.

MUKHERJEE, D. CROMLEY, R.G. SHAH, F.A. E BRAVO-URETA, B.E. Optimal location of centralized biodigesters for small dairy farms: A case study from the United States. **International Journal of sustainable energy planning and management**, v. 8, n. , 2015.

NANINJA, WISDOM. Optimizing Transportation Cost of Solid Waste:A Case Study in the Sunyani Municipality. , v. , n. , 2013.

ORLOFF, C. S.. A fundamental problem in vehicle routing. **Networks 4**, v. 4, n. 1, 1974.

OTOO, D., **Capacitated arc routing problem: collection of solid waste at Kwadaso estate**, 2012. Tese (Doutorado) - University of Energy and Natural Resources, Sunyani, Ghana.

PIMENTEL, R, **Planejamento do plantio e da colheita de cana-de-açucar utilizando técnicas matemáticas de otimização**, 2014. Tese (Doutorado em Agronomia) - Universidade Estadual Paulista, Botucatu, SP.

REINHARDT, L. B., PISINGER, D., MADSEN, O. B.G., AND KALLEHAUGE, B.. Routing and scheduling problems. , v. , n. , 2011.

RIPA, M. FIORENTINO, G. VACCA, V. E ULGIATI, S. The relevance of site-specific data in Life Cycle Assessment (LCA). The case of the municipal solid waste management in the metropolitan city of Naples (Italy). **Journal of Cleaner Production**, v. 142, n. , 2017.

RONEN, R., KELLERMAN, A. AND LAPIDOT, M.. Improvement of a solid waste collection system: the case of Givatayim, Israel. **Applied Geography**, v. 3, n. , 1983.

SOLOMON, M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations research**, v. 35, n. 2, 1987.

SONESSON, U.. Modelling of waste collection-a general approach to calculate fuel consumption and time. **Waste Management & Research**, v. 18, n. 2, 2000.

SULEMANA, A., DONKOR, E., FORKUO, K., AND ORDURO-KWARTENG, S.. Optimal routing of solid waste collection trucks: A review of methods. **Journal of Engineering**, v. 2018, n. , 2018.

TORO, O., ELIANA, M. ESCOBAR, Z., ANTONIO, H. AND GRANADA, E. ET AL.. Literature review on the vehicle routing problem in the green transportation context. **Luna Azul**, v. , n. 42, 2016.

TOTH, P. E VIGO, D., **The vehicle routing problem**. Bolonha, Itália: SIAM, 2002. 367 p. ISBN 0-89871-579-2.

VAN HIEP, N. E PRESTON, TR. Effect of cattle manure and biodigester effluent levels on growth and composition of water spinach. **Livestock Research for Rural Development**, v. 18, n. 4, 2006.

VILLAS BÔAS, ROBERTO L. ET AL. Efeito de doses e tipos de compostos orgânicos na produção de alface em dois solos sob ambiente protegido. **Horticultura Brasileira**, v. 22, n. , 2004.

VON POSER, I. E AWAD, A. R., Optimal routing for solid waste collection in cities by using real genetic algorithm*, In:* 2006 2nd International Conference on Information & Communication Technologies, 2006, Síria, p. 221-226.

WASSERMANN, G. BINNER, E. MOSTBAUER, P. E SALHOFER, S, Environmental relevance of landfills depending on different waste management strategies*, In:* Proceedings of Sardinia, 2005, Sardinia, p. .

APÊNDICE A – IMPLEMENTAÇÃO COMPUTACIONAL DO ALGORITMO DA HEURÍSTICA DE REDUÇÃO DE CUSTOS.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

int m, maxR, nR=0, N;
double mu, Q;

void Read(double *x, double *y, double *d, double *a, double *b, double *t, char
FileName[])
{
     int i, j;
     FILE *f;
     char buff[255];
     f = fopen(FileName,"r");
     for (i=0; i<4; i++)
          fgets(buff, 255, (FILE*) f);
     fscanf(f, "%d %lf", &m, &Q);
     for (i=6; i<10; i++)
          fgets(buff, 255, (FILE*) f);
     for (i=0; i<=N; i++)
          fscanf(f, "%d %lf %lf %lf %lf %lf %lf", &j, &x[i], &y[i], &d[i], &a[i], &b[i], &t[i]);
     }
     fclose(f);
}

void Distances(double **C, double *x, double *y)
{
     int i,j;
     for (i=0; i<=N; i++)
     {
          for (j=0; j<=i; j++)
          {
               C[i][j] = sqrt( pow(x[i]-x[j],2) + pow(y[i]-y[j],2) );
               C[j][i] = C[i][j];
          }
     }
}

void Savings(double **C, double **S, double *SS, int *Si, int *Sj)
{
     int i, j, k=0;
     for (i=1; i<=N; i++) //Savings are not calculated for the deposit itself
          for (j=i+1; j<=N; j++)
          {
               S[i][j] = C[i][0] + C[0][j] -mu*C[i][j];
```

```
                SS[k] = S[i][j];
                Si[k] = i;
                Sj[k] = j;
                k++;
            }
}

void swap(double *arr, int i, int j)
{
    double temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

void swap_int(int *arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

// C program for implementation of decreasing selection sort
void selectionSort(double *arr, int *Si, int *Sj, int n)
{
    int i, j, max_idx;
    // One by one move boundary of unsorted subarray
    for (i=0; i<n-1; i++)
    {
        // Find the máximum element in unsorted array
        max_idx = i;
        for (j=i+1; j<n; j++)
            if (arr[j] > arr[max_idx])
          max_idx = j;
        // Swap the found maximum element with the first element
        swap(arr, max_idx, i);
        swap_int(Si,  max_idx, i);
        swap_int(Sj,  max_idx, i);
    }
}

double CheckTime(double *a, double *b, double **C, double *s, double t, int i, int j)
{
    t = t + C[i][j];
    if (t<=a[j])
        t = a[j];
    if (t<=b[j])
        return(t+s[j]);
    else
        return(0.0);
}

void Add_Link(int I, int J, int *TR, int k, int n, int **R)
```

```
{
    int i;
    for (i=1; i<k; i++)
        TR[i] = R[n][i];
    TR[k]   = I;
    TR[k+1] = J;
    for (i=k+2; i<N+1; i++)
        TR[i] = R[n][i-1];
}

double Check_Route(double *a, double *b, double **C, double *d, double *s, int *TR)
{
    int i=1, L=0;
    double t=0.0;
    while (TR[i] != 0 && L <= Q)
    {
        L = L + d[TR[i]];
        t = t + C[TR[i-1]][TR[i]];
        if (t < a[TR[i]])
            t = a[TR[i]];
        if (t <= b[TR[i]])
            t = t + s[TR[i]];
        else
            return(0.0);
        i++;
    }
    t = t + C[TR[i-1]][TR[i]];
    return(t);
}
void reset_route(int *R) // Reset route to zeros
{
    int i;
    for (i=0; i<N+2; i++)
        R[i] = 0;
}

void Update_route(int *TR, int **R, int n)
{
    int i;
    for (i=1; i<N+2; i++)
        R[n][i] = TR[i];
}

void ClarkeWright(double *a, double *b, double **C, double *s, double **S, double
*SS, double *tR, double *x, double *y,
        double *d, int *IR, int *IR, int *Loads, int **R, int *Si, int *Sj, int *TR1, int *TR2)
{
    Distances(C, x, y);
    int k, l, L, n = 0, Ns;
    double t, t1, t2;
    Ns = N*(N-1)/2;
    Savings(C, S, SS, Si, Sj);
```

```
selectionSort(SS, Si, Sj, Ns);
while (n < maxR)
{
    l = 0;
    L = 0;
    t = 0.0;
    for (k=0; k<Ns; k++)
    {
        if (l == 0)
        {
            if (IR[Si[k]] == -1 && IR[Sj[k]] == -1)
            {
                Add_Link(Si[k], Sj[k], TR1, 1, n, R); // Add (i,j) to TR1 at pos 1
                Add_Link(Sj[k], Si[k], TR2, 1, n, R); // Add (j,i) to TR2 at pos 1
                t1 = Check_Route(a, b, C, d, s, TR1);
                t2 = Check_Route(a, b, C, d, s, TR2);
                if (t1 != 0)
                {
                    if (t2 != 0)
                    {
                        if (t1 < t2)
                        {
                            Update_route(TR1, R, n); // R[n] -> TR1
                            t = t1;
                        }
                        else
                        {
                            Update_route(TR2, R, n);  // R[n] -> TR2
                            t = t2;
                        }
                    }
                    else
                    {
                        Update_route(TR1, R, n); // R[n] -> TR1
                        t = t1;
                    }
                    IR[Si[k]] = n;
                    IR[Sj[k]] = n;
                    l = l + 2;
                    L = L + d[Si[k]] + d[Sj[k]];
                    nR++;
                }
                else
                {
                    if (t2!=0)
                    {
                        Update_route(TR2, R, n);  // R[n] -> TR2
                        t = t2;
                        IR[Si[k]] = n;
                        IR[Sj[k]] = n;
                        l = l + 2;
                        L = L + d[Si[k]] + d[Sj[k]];
```

```
                        nR++;
                    }
                }
            }
        }
        else
        {
            if (IR[Si[k]] == n && IR[Sj[k]] == n)
            {
            }
            else
            {
                if (R[n][1] == Si[k] && IR[Sj[k]] == -1)
                {
                    Add_Link(Sj[k], Si[k], TR1, 1, n, R); // Add (j,i) to TR1 at pos 1
                    t1 = Check_Route(a, b, C, d, s, TR1);
                    if (t1 != 0)
                    {
                        Update_route(TR1, R, n); // R[n] -> TR1
                        t = t1;
                        IR[Si[k]] = n;
                        IR[Sj[k]] = n;
                        l = l + 1;
                        L = L + d[Sj[k]];
                    }
                }
                else
                {
                    if (R[n][1] == Sj[k] && IR[Si[k]] == -1)
                    {
                        Add_Link(Si[k], Sj[k], TR1, 1, n, R); // Add (i,j) to TR1 at
pos 1
                        t1 = Check_Route(a, b, C, d, s, TR1);
                        if (t1 != 0)
                        {
                            Update_route(TR1, R, n); // R[n] -> TR1
                            t = t1;
                            IR[Si[k]] = n;
                            IR[Sj[k]] = n;
                            l = l + 1;
                            L = L + d[Si[k]];
                        }
                    }
                    else
                    {
                        if (R[n][IR[n]] == Si[k] && IR[Sj[k]] == -1)
                        {
                            Add_Link(Si[k], Sj[k], TR1, IR[n], n, R); // Add (i,j) to
TR1 at pos IR[n]
                            t1 = Check_Route(a, b, C, d, s, TR1);
                            if (t1 != 0)
                            {
```

```
                                    Update_route(TR1, R, n); // R[n] -> TR1
                                    t = t1;
                                    lR[Si[k]] = n;
                                    lR[Sj[k]] = n;
                                    l = l + 1;
                                    L = L + d[Sj[k]];
                                }
                            }
                            else
                            {
                                if (R[n][lR[n]] == Sj[k] && lR[Si[k]] == -1)
                                {
                                    Add_Link(Sj[k], Si[k], TR1, lR[n], n, R); // Add (j,i) to
TR1 at pos lR[n]
                                    t1 = Check_Route(a, b, C, d, s, TR1);
                                    if (t1 != 0)
                                    {
                                        Update_route(TR1, R, n); // R[n] -> TR1
                                        t = t1;
                                        lR[Si[k]] = n;
                                        lR[Sj[k]] = n;
                                        l = l + 1;
                                        L = L + d[Si[k]];
                                    }
                                }
                            }
                        }
                    }
                }
            }
            reset_route(TR1);
            reset_route(TR2);
        }
        lR[n] = l;
        tR[n] = t;
        Loads[n] = L;
        n++;
    }
    for (k=0; k<=N; k++)
        if (lR[k] == -1.0)
        {
            R[nR][1] = k;
            lR[k] = nR;
            lR[nR] = 1.0;
            Loads[nR] = d[k];
            tR[nR] = 2*C[0][k] + s[k];
            nR++;
        }
}

double Totalize(double *t)
```

```
{
    int i;
    double T=0.0;
    for (i=0; i<nR; i++)
        T+=t[i];
    return(T);
}

double TD(double **C, int *lR, int **R) // Total Distance
{
    int i,j;
    double f = 0.0;
    for (i=0; i<nR; i++)
        for (j=0; j<=lR[i]; j++)
            f += C[R[i][j]][R[i][j+1]];
    return(f);
}

char * export(int *lR, int **R, char Path[])
{
    int i,j;
    FILE *f;
    char target[255];
    strcpy(target, "routes_savings/");
    char * FileName = strtok(Path, "/");
    FileName = strtok(NULL, "/");
    strcat(target, FileName);
    f = fopen(target,"w+");
    for (i=0; i<nR; i++)
    {
        for (j=0; j<lR[i]+2; j++)
            fprintf(f, "%d,", R[i][j]+1); //Added +1 for sed or awk addressing
        fprintf(f, "\n");
    }
    fclose(f);
    return(FileName);
}

int main(int argc, char *argv[])
{
    if (argc != 4)
    {
        fprintf(stderr, "Usage %s <DatasetName> <EntriesNumber> <mu>\n",
argv[0]);
    exit(-1);
    }
    N = atoi(argv[2]);
    mu = atof(argv[3]);
    int i;
    double total_distance = 0.0;
    // Initialization
    double *x   = (double *) calloc (N+1, sizeof(double));
```

```
double *y   = (double *) calloc (N+1, sizeof(double));
double *a   = (double *) calloc (N+1, sizeof(double));
double *b   = (double *) calloc (N+1, sizeof(double));
double *s   = (double *) calloc (N+1, sizeof(double));
double *d   = (double *) calloc (N+1, sizeof(double));
double **C = (double **) calloc (N+1, sizeof(double *));
double **S = (double **) calloc (N+1, sizeof(double *));
int *IR =      (int *) calloc (N+1, sizeof(int));      // Node's route index
for (i=1; i<N+1; i++)
     IR[i] = -1;

for (i=0; i<=N; i++)
{
     C[i] = (double *) calloc (N+1, sizeof(double));
     S[i] = (double *) calloc (N+1, sizeof(double));
}

int Ns = N*(N-1)/2;
double *SS = (double *) calloc (Ns, sizeof(double));
int *Si =      (int *) calloc (Ns, sizeof(int));
int *Sj =      (int *) calloc (Ns, sizeof(int));

// Reading data
Read(x, y, d, a, b, s, argv[1]);

// More initialization
maxR = fmax(N,m);
int *lR    = (int *)     calloc (maxR, sizeof(int));            // Routes' lenght
int *L      = (int *)    calloc (maxR, sizeof(int));            // Routes' load
double *tR = (double *) calloc (maxR, sizeof(double)); // Routes' time
int **R    = (int **)    calloc (maxR, sizeof(int *));
for (i=0; i<maxR; i++)
     R[i] = (int *) calloc (N+2, sizeof(int));

int *TR1 = (int *) calloc (N+2, sizeof(int));   // Temp route 1
int *TR2 = (int *) calloc (N+2, sizeof(int)); // Temp route 2

// Processing
ClarkeWright(a, b, C, s, S, SS, tR, x, y, d, IR, lR, L, R, Si, Sj, TR1, TR2);

total_distance = TD(C, lR, R);

// Results
char * FileName = export(lR, R, argv[1]);
printf("%s\t%d\t%f\n",FileName,nR,total_distance);

// Freeing memory
free(x);
free(y);
free(a);
free(b);
free(s);
```

```
    free(d);
    for (i=0; i<=N; i++)
    {
        free(C[i]);
        free(S[i]);
    }
    free(IR);
    free(SS);
    free(Si);
    free(Sj);
    free(IR);
    free(tR);
    for (i=0; i<m; i++)
        free(R[i]);
    free(TR1);
    free(TR2);
    return(0);
}
```

APÊNDICE B – IMPLEMENTAÇÃO COMPUTACIONAL DO ALGORITMO DA HEURÍSTICA DE INSERÇÃO I1.

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<float.h>

int m, maxR, nR=0, N, crit;
double alpha1, alpha2, lambda, mu, Q;

void Read(double *x, double *y, double *d, double *a, double *b, double *t, char
FileName[])
{
     int i, j;
     FILE *f;
     char buff[255];
     f = fopen(FileName,"r");
     for (i=0; i<4; i++)
          fgets(buff, 255, (FILE*) f);
     fscanf(f, "%d %lf", &m, &Q);
     for (i=6; i<10; i++)
          fgets(buff, 255, (FILE*) f);
     for (i=0; i<=N; i++)
          fscanf(f, "%d %lf %lf %lf %lf %lf %lf", &j, &x[i], &y[i], &d[i], &a[i], &b[i], &t[i]);
     fclose(f);
}

void Distances(double **C, double *x, double *y)
{
     int i,j;
     for (i=0; i<=N; i++)
     {
          for (j=0; j<=i; j++)
          {
               C[i][j] = sqrt( pow(x[i]-x[j],2) + pow(y[i]-y[j],2) );
               C[j][i] = C[i][j];
          }
     }
}

void swap(double *arr, int i, int j)
{
     double temp = arr[i];
     arr[i] = arr[j];
     arr[j] = temp;
```

```
}

void swap_int(int *arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

// C program for implementation of decreasing selection sort
void selectionSort(double *arr, int *Si, int *Sj, int n)
{
    int i, j, max_idx;

    // One by one move boundary of unsorted subarray
    for (i=0; i<n-1; i++)
    {
        // Find the máximum element in unsorted array
        max_idx = i;
        for (j=i+1; j<n; j++)
            if (arr[j] > arr[max_idx])
      max_idx = j;

        // Swap the found maximum element with the first element
        swap(arr, max_idx, i);
        swap_int(Si,  max_idx, i);
        swap_int(Sj,  max_idx, i);
    }
}

void decreasingSort(double *arr, int *idx, int n)
{
    int i, j, max_idx;

    // One by one move boundary of unsorted subarray
    for (i=0; i<n-1; i++)
    {
        // Find the máximum element in unsorted array
        max_idx = i;
        for (j=i+1; j<n; j++)
            if (arr[j] > arr[max_idx])
      max_idx = j;

        // Swap the found maximum element with the first element
        swap(arr, max_idx, i);
        swap_int(idx, max_idx, i);
    }
}

void increasingSort(double *arr, int *idx, int n)
{
    int i, j, min_idx;
```

```
    // One by one move boundary of unsorted subarray
    for (i=0; i<n-1; i++)
    {
        // Find the máximum element in unsorted array
        min_idx = i;
        for (j=i+1; j<n; j++)
            if (arr[j] < arr[min_idx])
      min_idx = j;

        // Swap the found maximum element with the first element
        swap(arr, min_idx, i);
        swap_int(idx, min_idx, i);
    }
}

double CheckTime(double *a, double *b, double **C, double *s, double t, int i, int j)
{
    t = t + C[i][j];
    if (t<=a[j])
        t = a[j];
    if (t<=b[j])
        return(t+s[j]);
    else
        return(0.0);
}

void reset_route(int *R) // Reset route to zeros
{
    int i;
    for (i=0; i<N+2; i++)
        R[i] = 0;
}

void Add_Link(int I, int J, int *TR, int k, int n, int **R)
{
    int i;
    reset_route(TR);
    for (i=1; i<k; i++)
        TR[i] = R[n][i];
    TR[k]   = I;
    TR[k+1] = J;
    for (i=k+2; i<N+1; i++)
        TR[i] = R[n][i-1];
}

double Check_Route(double *a, double *b, double **C, double *d, double *s, int *TR)
{
    int i=1, L=0;
    double t=0.0;
    while (TR[i] != 0)
    {
```

```
            L = L + d[TR[i]];
            t = t + C[TR[i-1]][TR[i]];
            if (t < a[TR[i]])
                t = a[TR[i]];
            if (t <= b[TR[i]] && L <= Q)
                t = t + s[TR[i]];
            else
                return(0.0);
            i++;
      }
      t = t + C[TR[i-1]][TR[i]];
      return(t);
}

void Update_route(int *TR, int **R, int n)
{
      int i;
      for (i=1; i<N+2; i++)
            R[n][i] = TR[i];
}

void print_route(int n, int **R)
{
      int i=0;
      while(R[n][i] != 0 || i == 0)
      {
            printf("%d\t", R[n][i]);
            i++;
      }
      printf("\n\n");
}

void print_tmp_route(int *TR)
{
      int i;
      printf("\n");
      for(i=0; i<=N; i++)
      {
            printf("%d\t", TR[i]);
            if (TR[i] == 0 && i != 0)
                  break;
      }
      printf("\n\n");
}

double compute_W(double *a, double **C, int j, int *TR)
{
 int i;
      double t = 0.0;
 for (i=1; i<=N; i++)
      {
            t = t + C[TR[i-1]][TR[i]];
```

```
        if (t < a[TR[i]])
            t = a[TR[i]];
        if (TR[i] == j)
            break;
    }
    return(t);
}

void compute_C(double *a, double **C, double **C1, double **C2, double **c11,
double **c12, int **FM, int n, int **R, int *TR1, int *TR2)
{
    int i, j, k, u;
    double Wj, Wju;
    for (u=1; u<=N; u++)    // Node loop
        for (k=1; k<=N; k++) // Position in-route loop
        {
            if (FM[u][k] == 1)
            {
                i = R[n][k-1];
                j = R[n][k];
                c11[u][k] = C[i][u] + C[u][j] - mu*C[i][j];
                Add_Link(i, j, TR1, k, n, R); // Add ( i, j ) to TR1 at pos k
                Add_Link(u, j, TR2, k, n, R); // Add ( u, j ) to TR1 at pos k
                Wj  = compute_W(a, C, j, TR1);
                Wju = compute_W(a, C, j, TR2);
                c12[u][k] = Wju - Wj;
                C1[u][k] = alpha1*c11[u][k] + alpha2*c12[u][k];
                C2[u][k] = lambda*C[0][u] - C1[u][k];
            }
        }
}

void reset_int_matrix(int **M, int m, int n) // Set an integer m*n matrix to zeros
{
    int i,j;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            M[i][j] = 0;
}

void reset_dbl_matrix(double **M, int m, int n) // Set an integer m*n matrix to zeros
{
    int i,j;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            M[i][j] = 0.0;
}

int min(double **C, int **FM)
{
    int i, j, l;
    double min=DBL_MAX;
```

```
    for (i=0; i<=N; i++)
        for (j=0; j<=N; j++)
            if (C[i][j] <= min && FM[i][j] == 1)
            {
                min = C[i][j];
                l = j;
            }
    return(l);
}

int max(double **C, int **FM, int j)
{
    int i, k;
    double max=-DBL_MAX;
    for (i=0; i<=N; i++)
        if (C[i][j] >= max && FM[i][j] == 1)
        {
            max = C[i][j];
            k = i;
        }
    return(k);
}

void Insertion(double *a, double *b, double **C, double **C1, double **C2, double
**c11, double **c12, double *d, double *dummy, double *s, double *tR, double *x,
double *y, int *CR, int **FM, int *IR, int *lR, int *Loads, int **R, int *TR1, int *TR2)
{
    int i, j, k, l, L, n=0, node, opt, posi, routed=0;
    double t, t1, t2;
    Distances(C, x, y);
    if (crit == 1) // Farthest unrouted node
    {
        for (i=0; i<N+1; i++)
            dummy[i] = C[0][i];
        decreasingSort(dummy, CR, N+1);
    }
    else // Earliest unrouted node
    {
        for (i=0; i<N+1; i++)
            dummy[i] = b[i];
        increasingSort(dummy, CR, N+1);
    }
    while (n<maxR && routed<N)
    {
        l = 0;
        L = 0;
        t = 0.0;
        for (k=0; k<=N; k++)
        {
            reset_route(TR1);
            reset_route(TR2);
            if (l == 0)
```

```
            {
                i = 0;
                j = 0;
                while (j == 0)
                {
                    if (IR[CR[i]] == -1)
                    {
                        TR1[1] = CR[i];
                        j++;
                        break;
                    }
                    i++;
                }
                t1 = Check_Route(a, b, C, d, s, TR1);
                Update_route(TR1, R, n); // R[n] -> TR1
                routed++;
                t = t1;
                IR[CR[i]] = n;
                l = l + 1;
                L = L + d[CR[i]];
                nR++;
            }
            //if (l != 0)
            else
        {
                // Compute factibility matrix
                opt = 0;
                for (i=1; i<=l; i++)   // Position in-route loop
                {
                    for (j=1; j<=N; j++) //     Node loop
                    {
                        if (IR[j] == -1)
                        {
                            Add_Link(j, R[n][i], TR1, i, n, R); // Add ( j, R[n][i] ) to TR1
at pos i
                            Add_Link(R[n][i], j, TR2, i, n, R); // Add ( R[n][i], j ) to TR2
at pos i
                            t1 = Check_Route(a, b, C, d, s, TR1);
                            t2 = Check_Route(a, b, C, d, s, TR2);
                            if (t1 != 0)
                            {
                                FM[j][i] = 1;
                                opt++;
                            }
                            if (t2 != 0)
                            {
                                FM[j][i+1] = 1;
                                opt++;
                            }
                        }
                    }
                }
```

```
                    if (opt == 0)
                        break;
                    else // Compute C1 & C2
                    {
                        compute_C(a, C, C1, C2, c11, c12, FM, n, R, TR1, TR2);
                        posi = min(C1, FM);
                     node = max(C2, FM, posi);
                        Add_Link(node, R[n][posi], TR1, posi, n, R); // Add ( node, R[n]
[posi] ) to TR1 at posi
                        t1 = Check_Route(a, b, C, d, s, TR1);
                        Update_route(TR1, R, n); // R[n] -> TR1
                        routed++;
                        t = t1;
                        lR[node] = n;
                        l = l + 1;
                        L = L + d[node];

                        reset_dbl_matrix(c11, N+1, N+1);
                        reset_dbl_matrix(c12, N+1, N+1);
                        reset_dbl_matrix(C1, N+1, N+1);
                        reset_dbl_matrix(C2, N+1, N+1);

                        reset_int_matrix(FM, N+1, N+1);
                    }
                }
            }
            lR[n] = l;
            tR[n] = t;
            Loads[n] = L;
            n++;
        }
}

double Totalize(double *t)
{
    int i;
    double T=0.0;
    for (i=0; i<nR; i++)
        T+=t[i];
    return(T);
}

double TD(double **C, int *lR, int **R) // Total Distance
{
    int i,j;
    double f = 0.0;
    for (i=0; i<nR; i++)
        for (j=0; j<=lR[i]; j++)
            f += C[R[i][j]][R[i][j+1]];
    return(f);
}
```

```
char * export(int *lR, int **R, char Path[])
{
    int i,j;
    FILE *f;
    char target[255];
    strcpy(target, "routes_insertion/");
    char * FileName = strtok(Path, "/");
    FileName = strtok(NULL, "/");
    strcat(target, FileName);
    f = fopen(target,"w+");
    for (i=0; i<nR; i++)
    {
        for (j=0; j<lR[i]+2; j++)
            fprintf(f, "%d,", R[i][j]+1); //Added +1 for sed or awk addressing
        fprintf(f, "\n");
    }
    fclose(f);
    return(FileName);
}

int main(int argc, char *argv[])
{
    if (argc != 8)
    {
        fprintf(stderr, "Usage %s <DatasetName> <EntriesNumber> <Criteria: 1 or 2>
<alpha_1> <alpha_2> <lambda> <mu>\n", argv[0]);
    exit(-1);
    }
    N = atoi(argv[2]);
    crit = atoi(argv[3]);
    alpha1 = atof(argv[4]);
    alpha2 = atof(argv[5]);
 lambda = atof(argv[6]);
    mu = atof(argv[7]);
    int i;
    double total_distance = 0.0;
    // Initialization
    double *x   = (double *) calloc (N+1, sizeof(double));
    double *y   = (double *) calloc (N+1, sizeof(double));
    double *a   = (double *) calloc (N+1, sizeof(double));
    double *b   = (double *) calloc (N+1, sizeof(double));
    double *s   = (double *) calloc (N+1, sizeof(double));
    double *d   = (double *) calloc (N+1, sizeof(double));

    double *dummy = (double *) calloc (N+1, sizeof(double)); // Distance to node 0 or
earliest service time

    double **C = (double **) calloc (N+1, sizeof(double *));
    double **S = (double **) calloc (N+1, sizeof(double *));
    int **FM   = (int **)    calloc (N+1, sizeof(int *));

    double **c11 = (double **) calloc (N+1, sizeof(double *)); // c11
```

```
    double **c12 = (double **) calloc (N+1, sizeof(double *)); // c12
    double **C1  = (double **) calloc (N+1, sizeof(double *)); // c1
    double **C2  = (double **) calloc (N+1, sizeof(double *)); // c2

    int *IR = (int *) calloc (N+1, sizeof(int));  // Node's route index
    int *CR = (int *) calloc (N+1, sizeof(int));  // Criteria index


    for (i=1; i<N+1; i++)
    {
        IR[i] = -1;
        CR[i] = i;
    }

    for (i=0; i<=N; i++)
    {
        C[i]  = (double *) calloc
(N+1, sizeof(double));
        S[i]  = (double *) calloc (N+1, sizeof(double));
        FM[i] = (int *)   calloc (N+1, sizeof(int));

        c11[i] = (double *) calloc (N+1, sizeof(double));
        c12[i] = (double *) calloc (N+1, sizeof(double));
        C1[i]  = (double *) calloc (N+1, sizeof(double));
        C2[i]  = (double *) calloc (N+1, sizeof(double));
    }

    // Reading data
    Read(x, y, d, a, b, s, argv[1]);

    // More initialization
    maxR = fmax(N,m);
    int *lR   = (int *)   calloc (maxR, sizeof(int));          // Routes' lenght
    int *L    = (int *)   calloc (maxR, sizeof(int));          // Routes' load
    double *tR = (double *) calloc (maxR, sizeof(double)); // Routes' time
    int **R   = (int **)  calloc (maxR, sizeof(int *));
    for (i=0; i<maxR; i++)
        R[i] = (int *) calloc (N+2, sizeof(int));

    int *TR1 = (int *) calloc (N+2, sizeof(int));  // Temp route 1
    int *TR2 = (int *) calloc (N+2, sizeof(int)); // Temp route 2

    // Processing
    Insertion(a, b, C, C1, C2, c11, c12, d, dummy, s, tR, x, y, CR, FM, IR, lR, L, R,
TR1, TR2);

    total_distance = TD(C, lR, R);

    // Results
    char * FileName = export(lR, R, argv[1]);
    printf("%s\t%d\t%f\n",FileName,nR,total_distance);
```

```
    // Freeing memory
    free(x);
    free(y);
    free(a);
    free(b);
    free(s);
    free(d);
    free(dummy);
    for (i=0; i<=N; i++)
    {
        free(C[i]);
        free(S[i]);
        free(FM[i]);
        free(c11[i]);
        free(c12[i]);
        free(C1[i]);
        free(C2[i]);
    }
    free(IR);
    free(CR);

    free(lR);
    free(tR);
    for (i=0; i<maxR; i++)
        free(R[i]);
    free(TR1);
    free(TR2);
    return(0);
}
```