Centro Federal de Educação Tecnológica de Minas Gerais
Programa de Pós-Graduação em Modelagem Matemática e Computacional

Algoritmos *Iterated Local Search* e *Simulated Annealing* Aplicados ao Problema de Localização com Cobertura Parcial

Leonardo Correa Cardoso

Belo Horizonte
Julho de 2023

Leonardo Correa Cardoso

Algoritmos *Iterated Local Search* e *Simulated Annealing* Aplicados ao Problema de Localização com Cobertura Parcial

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, como parte dos requisitos para obtenção do título de Mestre em Modelagem Matemática e Computacional.

Área de Concentração: Modelagem Matemática e Computacional

Linha de Pesquisa: Sistemas Inteligentes

Orientadora: Profa . Dra. Elisangela Martins de Sá

Coorientador: Prof. Dr. Sérgio Ricardo de Souza

Belo Horizonte - MG
Julho de 2023

Cardoso, Leonardo Correa
C268a Algorítmos iterated local search e simulated annealing aplicados ao problema de localização com cobertura parcial / Leonardo Correa Cardoso. – 2023.
61 f.

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientadora: Elisângela Martins de Sá.
Coorientador: Sérgio Ricardo de Souza.
Dissertação (mestrado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Problema de localização – Teses. 2. Cobertura parcial – Teses. 3. Metaheurística – Teses. 4. Pesquisa local iterada – Teses. 5. Recozimento simulado (Matemática) – Teses. I. Sá, Elisangela Martins de. II. Souza, Sérgio Ricardo de. III. Centro Federal de Educação Tecnológica de Minas Gerais. IV. Título.

CDD 519.6

Elaboração da ficha catalográfica pela bibliotecária Jane Marangon Duarte, CRB 6º 1592 / Cefet/MG

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO CURSO DE MESTRADO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

"ALGORITMOS ITERATED LOCAL SEARCH E SIMULATED ANNEALING APLICADOS AO PROBLEMA DE LOCALIZAÇÃO COM COBERTURA PARCIAL"

Dissertação de Mestrado apresentada por **Leonardo Correa Cardoso**, em 14 de julho de 2023, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Profª. Drª. Elisangela Martins de Sá
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Rodney Oliveira Marinho Diana
Banco de Desenvolvimento de Minas Gerais

Prof. Dr. Gustavo Campos Menezes
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida a impressão,

Prof. Dr. Thiago de Souza Rodrigues
Coordenador do Programa de Pós-Graduação em
Modelagem Matemática e Computacional

Dedico este trabalho à memória do meu
avô Professor Derbe Correa, que nos
deixou há pouco tempo, mas fez tanto
por mim ao longo da sua vida.

Agradecimentos

Ao finalizar este trabalho, tenho a certeza de que é importante agradecer a todos que contribuíram para mais essa realização.

Primeiramente a Deus, por me permitir superar os diversos obstáculos encontrados ao longo da realização deste trabalho.

A minha orientadora, Profa. Dra. Elisangela Martins de Sá, agradeço pela oportunidade, pela paciência, aprendizado compartilhado e presença em todos os momentos que tive nessa caminhada.

Ao meu Coorientador, Prof. Dr. Sérgio Ricardo de Souza, por todo o conhecimento e incentivo. Agradeço a ele antes mesmo de ingressar no programa, pela orientação e direcionamento para que fosse possível tornar esse sonho realidade.

Aos meus pais, Silvia e Amarildo, pelo educação e apoio incondicional durante toda a minha vida. Deixo aqui minha eterna gratidão por tudo.

Ao meu irmão João Paulo, pela amizade, pelas intermináveis conversas e por ser uma pessoa na qual eu sempre poderei confiar.

A minha esposa Bárbara, pelo companheirismo, agradeço principalmente pela paciência e apoio nos momentos mais difíceis.

Aos amigos e professores do PPGMMC por todo conhecimento compartilhado, em particular, gostaria de agradecer ao Fábio Mourão pela amizade e todo conhecimento compartilhado.

A todas as demais pessoas que contribuíram direta e indiretamente para a realização deste trabalho.

Ao CEFET-MG, que nos forneceu materiais e condições para desenvolver esta pesquisa.

Resumo

O Problema de Localização com Cobertura Parcial consiste em localizar um conjunto de instalações de forma a minimizar o custo total de localização e garantir que uma quantidade predeterminada de demanda de clientes seja coberta por estas instalações. Este trabalho apresenta dois algoritmos para a resolução deste problema, sendo o primeiro baseado na meta-heurística *Iterated Local Search* e o segundo baseado na meta-heurística *Simulated Annealing*. Além disso, um conjunto de experimentos computacionais foram realizados e resultados demonstram que boas soluções podem ser encontradas para instâncias moderadamente grandes.

Palavras-chave: Problema de Localização com Cobertura Parcial, Meta-heurística, *Iterated Local Search*, *Simulated Annealing*, Problemas de Localização.

Abstract

The Location Problem with Partial Coverage consists of locating a set of facilities in such a way as to minimize the total location cost and ensure that a predetermined amount of customer demand is covered by these facilities. This work presents two algorithms to solve this problem: the first one is based on the Iterated Local Search metaheuristic, and the second one is based on the Simulated Annealing metaheuristic. Furthermore, a series of computational experiments were conducted, and the results demonstrate that good solutions can be found even for moderately large instances.

Keywords: Partial Set Covering Location Problem, Meta-heuristics, Iterated Local Search, Simulated Annealing, Location Problem.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Capítulo 1

Introdução

Problemas de localização de cobertura fazem parte de um grupo de problemas de localização de grande importância. Estes problemas consistem em determinar a localização de instalações de forma a cobrir demandas de clientes. Neste tipo de problemas, em geral, um cliente é considerado coberto por uma instalação se a distância entre os dois for menor ou igual a uma distância máxima previamente definida, denominada raio de cobertura.

Na literatura, encontram-se diversas aplicações para problemas de cobertura. Church & ReVelle (1974) propuseram o Problema de Máxima Cobertura (PMC) – *Maximal Covering Location Problem*. Este problema tem como objetivo maximizar a cobertura de uma certa população, considerando um número fixo de instalações candidatas. Os autores apresentaram ainda uma curva de variação, analisando a relação entre o número de instalações elegíveis para fornecer cobertura e a quantidade da população coberta à medida que adiciona-se novas instalações à solução corrente. Galvão & ReVelle (1996) apresentam uma heurística lagrangeana para o mesmo problema, além de resultados de testes computacionais para redes com até 150 nós. Mišković (2017) propõe uma variação robusta para o problema de cobertura máxima dinâmica – *Dynamic Maximal Covering Location Problem*. O problema é resolvido através da meta-heurística *Variable Neighborhood Search - Linear Program* (VNS-LP), baseada na hibridização de uma busca em vizinhança variável e uma técnica de programação linear. Coco et al. (2018) apresentam uma abordagem robusta para lidar com a incerteza nos dados de entrada do PMC.

Novas variações para problemas de cobertura foram introduzidas por Daskin & Owen (1999). Eles apresentaram o Problema p-Centro de Cobertura Parcial, que minimiza a distância de cobertura de tal forma que uma determinada fração da população é coberta. Foi apresentado também o Problema de Localização de Cobertura de Conjunto Parcial – *Partial Set Covering location Problem* (PSCLP), que busca minimizar o custo de abertura das instalações necessárias para cobrir uma certa fração

da população. Nesse trabalho, os autores tiveram como objetivo preencher a lacuna que até então existia, ao propor modelos que podem servir de base para um processo de planejamento logístico no qual pode ser impossível ou antieconômico atender todos os clientes igualmente bem. Além disso, pode-se identificar locais com clientes que são particularmente difíceis de serem atendidos.

Esta dissertação tem como foco o PSCLP. Este problema também foi abordado por Bilal et al. (2014), que propõem uma heurística *Iterated Tabu Search* para uma variação do problema que tem como objetivo maximizar o lucro, porém não sendo necessário cobrir toda a demanda. O algoritmo proposto usa dois níveis de perturbação e uma meta-heurística de Busca Tabu. Cordeau et al. (2019) apresentam uma abordagem de resolução para o PSCLP utilizando o método de decomposição de Benders com foco em reduzir tempos computacionais para a resolução de instâncias de grandes dimensões do problema.

Como o PSCLP é um problema NP-difícil, para resolver instâncias de grande porte do problema o uso de procedimentos heurísticos é uma alternativa muito utilizada. Portanto, esta dissertação propõe dois algoritmos. O primeiro algoritmo é baseado na meta-heurística *Iterated Local Search* (ILS) e o segundo algoritmo é baseado na meta-heurística *Simulated Annealing* (SA). A partir de experimentos computacionais utilizando o conjunto de instâncias propostas por Cordeau et al. (2019), é feita uma comparação dos tempos de execução e da qualidade das soluções obtidos pelos algoritmos propostos e pelo *solver* CPLEX.

1.1 Justificativa

Conforme exposto por Cordeau et al. (2019), os problemas de cobertura de conjuntos parciais possuem uma grande relevância prática, porém a sua abordagem na literatura científica ainda é limitada. Pode-se citar sua aplicação em diversos processos de tomadas de decisão recorrentes na sociedade atual, como dimensionamento de pontos para instalação de antenas com tecnologia 5G, pontos de recarga para veículos elétricos, além de outras abordagens já usuais, como a instalação de pontos de distribuição logística, dentre outras.

Todas estas abordagens possuem uma característica em comum que as tornam diferente dos demais problemas de cobertura. No problema de cobertura parcial não existe a necessidade de cobrir 100% das demandas dos clientes. Essa é uma característica que se aproxima de projetos reais, nos quais muitas vezes não é necessário realizar a cobertura integral dos clientes, sendo necessário apenas o atendimento de uma determinada parte da demanda total. Por outro lado, é fundamental selecionar as instalações de maneira a minimizar os seus custos de abertura. Com isso, determi-

nar qual o melhor local para ser instalada as instalações com o menor custo possível e ainda atendendo à demanda solicitada pode se tornar um problema complexo, à medida que se aumenta a quantidade de clientes e de candidatas a instalações.

Desse modo, surge a necessidade de se desenvolver ferramentas capazes de localizar as instalações com maior assertividade possível. Uma vez que os métodos exatos não conseguem tratar instâncias de dimensões elevadas com bons tempos computacionais, os algoritmos meta-heurísticos são técnicas promissoras para problemas combinatórios, pois, apesar das mesmas não garantirem a obtenção de soluções ótimas, é possível obter boas soluções em um tempo computacional adequado.

1.2 Objetivos Geral e específicos

O objetivo geral do trabalho é propor heurísticas eficientes para selecionar, em um tempo computacional satisfatório, um conjunto de instalações com o menor custo de abertura possível, de modo a cobrir uma quantidade de clientes, atendendo a uma determinada demanda mínima.

Os objetivos específicos a serem atingidos são:

(i) Estudar os principais trabalhos da literatura que abordam o problemas de localização envolvendo cobertura;

(ii) Desenvolver algoritmos meta-heurísticos que gerem boas soluções para o problema;

(iii) Executar experimentos computacionais utilizando os algoritmos propostos e o *solver* CPLEX;

(iv) Fazer uma análise estatística dos resultados obtidos;

(v) Comparar os resultados dos algoritmos desenvolvidos entre si e com os resultados do solver CPLEX.

1.3 Metodologia de Pesquisa

Para se obter os objetivos propostos, esta dissertação foi desenvolvido com a seguinte metodologia:

(i) **Revisão bibliográfica**: Pesquisar os principais artigos que tratam o problema. Observar, nos artigos da literatura, quais técnicas foram utilizadas para solução dos problemas propostos;

(ii) **Descrição do problema**: Realizar uma descrição formal do problema, apresentando uma formulação matemática que represente as suas características;

(iii) **Definição de abordagens**: Definir as abordagens utilizadas para buscar boas soluções em tempo computacional viável;

(iv) **Implementação de algoritmos meta-heurísticos**: Após analisar as técnicas heurísticas para buscar soluções para problemas combinatórios, desenvolver algoritmos para o problema proposto;

(v) **Análise dos resultados obtidos**: Para compreender e validar os resultados obtidos, realizar análises estatísticas dos resultados e comparar os mesmos com resultados do otimizador CPLEX.

1.4 Organização do trabalho

Os demais capítulos desta dissertação estão organizados da seguinte maneira. O Capítulo 2 apresenta formalmente o problema principal e os demais problemas relacionados a ele. O Capítulo 3 descreve-se os trabalhos da literatura relacionados ao tema. No Capítulo 4 é apresentada a fundamentação teórica. O Capítulo 5 descreve a metodologia utilizada para a solução do modelo matemático. O Capítulo 6 apresenta os resultados dos experimentos computacionais obtidos após a implementação e testes com um conjunto de instâncias da literatura. Por fim, o Capítulo 7 apresenta as conclusões obtidas e propostas para trabalhos futuros.

Capítulo 2

Descrição do Problema

Este capítulo apresenta uma descrição do Problema de Localização com Cobertura Parcial. Inicialmente, é apresentada uma introdução aos problemas de cobertura de conjuntos na Seção 2.1. Em seguida, na Seção 2.2 é apresentado o problema de localização de cobertura máxima. Após isso, na Seção 2.3, é feita a apresentação do Problema de Localização com Cobertura Parcial, problema central deste trabalho, evidenciando as características que o define em meio a problemas similares na literatura.

2.1 Problema de Cobertura de Conjuntos

Conforme apresentado no trabalho de Farahani et al. (2012), o problema de cobertura de conjuntos foi introduzido inicialmente por Hakimi (1965), que apresenta um modelo que visa determinar o número mínimo de policiais necessários para cobrir uma rede de rodovias. O problema é resolvido através da geração de todas as coberturas de vértices de um grafo por meio de uma função booleana definida sobre os vértices de um grafo. Já a primeira formulação matemática de um problema de cobertura de conjuntos foi apresentado por Toregas et al. (1971), com base em um modelo que busca minimizar o número de instalações de serviços de emergência, levando em consideração os aspectos de tempo e distância máxima que separam um usuário do serviço mais próximo.

O modelo proposto resumido é apresentado a seguir, com base na seguinte notação. Seja I o conjunto formado por n instalações candidatas e J um conjunto com m clientes. Uma representação das possíveis coberturas de clientes por instalações é dada através de uma matriz de cobertura $A_{n\times m}$. Esta matriz possui elementos $a_{ij} \in \{0, 1\}$, em que $a_{ij} = 1$ caso o cliente j esteja dentro do raio de cobertura, denotado por R, da instalação i, e $a_{ij} = 0$, caso contrário.

As variáveis do modelo matemático são y_i, que é uma variável binária tal que:

$$y_i = \begin{cases} 1, & \text{se a instalação } i \in I \text{ estiver aberta} \\ 0, & \text{caso contrário.} \end{cases}$$

A formulação matemática para o problema de cobertura de conjuntos, baseada em Toregas et al. (1971), é dada por:

$$\min \sum_{i \in I} y_i \tag{2.1}$$

$$\sum_{i \in N} a_{ij} y_i \geq 1 \qquad j \in J \tag{2.2}$$

$$y_i \in \{0, 1\} \qquad i \in I. \tag{2.3}$$

A função objetivo (2.1) minimiza o número total de instalações abertas. As restrições (2.2) apresentam a garantia de atendimento de cada nó de demanda por pelo menos uma instalação e as restrições (2.3) garantem a integralidade das variáveis de cobertura.

2.2 Problema de Localização de Cobertura Máxima

Introduzido por Church & ReVelle (1974) e reformulado por Cordeau et al. (2019), o problema de localização de máxima cobertura (PMC) consiste em determinar a localização de um conjunto de instalações, de forma a cobrir (atender) a maior parcela de uma determinada população ou de clientes dentro de um raio de cobertura R, com um orçamento limitado. Os nós da rede referentes a clientes que estiverem a uma distância de algum nó escolhido para abrir uma instalação que seja menor ou igual ao raio de cobertura são classificados como cobertos. Neste caso, a população ou demanda associada a esses nós é considerada atendida.

O modelo adotado para o PMC utiliza a seguinte notação. Seja I o conjunto formado por n instalações candidatas e J um conjunto com m clientes. Cada cliente $j \in J$ possui uma demanda d_j e cada instalação $i \in I$ possui um custo de abertura f_i. Existe, também, um capital máximo disponível, dado por B.

Uma representação das possíveis coberturas de clientes por instalações é dada por uma matriz de cobertura $A_{n \times m}$. Esta matriz possui elementos $a_{ij} \in \{0, 1\}$, em que $a_{ij} = 1$, caso o cliente j esteja dentro do raio de cobertura R da instalação i, e $a_{ij} = 0$, caso contrário.

As variáveis do modelo matemático são y_i, que é uma variável binária tal que:

$$y_i = \begin{cases} 1, & \text{se a instalação } i \in I \text{ estiver aberta;} \\ 0, & \text{caso contrário.} \end{cases}$$

e z_j, que é uma variável binária tal que:

$$z_j = \begin{cases} 1, & \text{se o cliente } j \in J \text{ for coberto por pelo menos uma instalação;} \\ 0, & \text{caso contrário.} \end{cases}$$

O modelo matemático para o problema, conforme Cordeau et al. (2019), é dado por:

$$\max \sum_{i=1}^{m} d_j z_j \tag{2.4}$$

$$s.a \sum_{i=1}^{n} a_{ij} y_i \geq z_j \qquad \forall j \in J \tag{2.5}$$

$$\sum_{j=1}^{n} f_i y_i \leq B \tag{2.6}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{2.7}$$

$$z_j \in \{0,1\} \qquad \forall j \in J. \tag{2.8}$$

A função objetivo (2.4) maximiza o atendimento das demandas. As restrições (2.5) garantem que, caso um cliente j esteja coberto, haverá pelo menos uma instalação aberta dentro do seu raio de cobertura o cobrindo. A restrição (2.6) garante que o orçamento disponível para a abertura das instalações não seja excedido. Finalmente, as restrições (2.7) e (2.8) impõem restrições binárias às variáveis de decisão y e z, respectivamente.

2.3 Problema de Localização de Cobertura de Conjunto Parcial

O problema de localização com cobertura parcial tem como objetivo minimizar o custo de abertura de novas instalações, ao mesmo tempo em que deve garantir o atendimento de uma demanda mínima.

Seja I o conjunto formado por n instalações candidatas e J um conjunto com m clientes. Cada cliente $j \in J$ possui uma demanda d_j e cada instalação $i \in I$ possui um custo de abertura f_i. Existe também uma demanda mínima a ser atendida, dada

por D. Uma representação das possíveis coberturas de clientes por instalações é dada através de uma matriz de cobertura $A_{n\times m}$, que possui elementos $a_{ij} \in \{0,1\}$, em que $a_{ij} = 1$, caso o cliente j esteja dentro do raio de cobertura R da instalação i, e $a_{ij} = 0$, caso contrário.

As variáveis do modelo matemático são y_i, que é uma variável binária tal que:

$$y_i = \begin{cases} 1, & \text{se a instalação } i \in I \text{ estiver aberta;} \\ 0, & \text{caso contrário,} \end{cases}$$

e z_j, que é uma variável binária tal que:

$$z_j = \begin{cases} 1, & \text{se o cliente } j \in J \text{ for coberto por pelo menos uma instalação;} \\ 0, & \text{caso contrário.} \end{cases}$$

Pode-se formular o PSCLP, conforme descrito por Cordeau et al. (2019), da seguinte forma:

$$\min \sum_{i=1}^{n} f_i y_i \tag{2.9}$$

$$s.a \sum_{i=1}^{n} a_{ij} y_i \geq z_j \qquad \forall j \in J \tag{2.10}$$

$$\sum_{j=1}^{m} d_j z_j \geq D \tag{2.11}$$

$$y_i \in \{0,1\} \qquad \forall i \in I \tag{2.12}$$

$$z_j \in \{0,1\} \qquad \forall j \in J. \tag{2.13}$$

A função objetivo (2.9) minimiza o custo de abertura das instalações. As restrições (2.10) garantem que, caso um cliente j esteja coberto, haverá pelo menos uma instalação aberta dentro do seu raio de cobertura o cobrindo. A restrição (2.11) garante que o mínimo de demanda D seja atendido. Finalmente, as restrições (2.12) e (2.13) impõem restrições binárias às variáveis de decisão y e z, respectivamente.

2.3.1 Instância ilustrativa

Um exemplo para o problema de cobertura parcial de conjuntos será dado a seguir. Considere custos de abertura de 4 instalações candidatas f_i e as demandas d_j de 8

clientes, dados, respectivamente, pelos vetores:

$$\begin{aligned} f &= [10, 15, 5, 30] \\ d &= [20, 15, 50, 30, 10, 5, 30, 10]. \end{aligned}$$

Para o exemplo proposto, a distância entre os clientes e instalações em potencial é dada por uma matriz de distâncias:

$$\Delta_{n\times m} = \begin{pmatrix} 2 & 3 & 4 & 6 & 7 & 8 & 11 & 13 \\ 5 & 4 & 4 & 3 & 5 & 7 & 8 & 9 \\ 6 & 8 & 9 & 5 & 3 & 4 & 4 & 7 \\ 12 & 14 & 10 & 7 & 4 & 2 & 2 & 1 \end{pmatrix}.$$

Considere ainda, para o exemplo proposto, que a relação de cobertura é dada por um raio de cobertura $R = 4$. Então, a matriz binária de cobertura $A_{n\times m}$, em que $a_{ij} = 1$, caso o cliente j esteja dentro do raio de cobertura R da instalação candidata i, e $a_{ij} = 0$, caso contrário, é dada por:

$$A_{n\times m} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

A Figura 2.1 ilustra as possíveis relações de cobertura através da disponibilidade de 4 instalações candidatas em potencial e 8 clientes a serem atendidos conforme descrito previamente.

Para o exemplo, podemos somar o vetor d de demandas e, assim, determina-se a demanda total, que é igual a 170. Porém, o problema de cobertura parcial de conjuntos caracteriza-se por ter a obrigação de cobrir pelo menos um certo percentual da demanda total de clientes. Consideremos, então, que deve ser coberta 50% da demanda total; assim, $D = 85$. Cobrindo essa demanda, podemos representar um solução viável para o problema através dos vetores binários y de instalações ativas e z de clientes atendidos, conforme a seguir:

$$\begin{aligned} y &= [1, 0, 0, 0] \\ z &= [1, 1, 1, 0, 0, 0, 0, 0]. \end{aligned}$$

A solução apresentada tem o valor de função objetivo dado por $fo = 10$ e uma demanda total coberta igual a $D = 85$, que, portanto, atende todas as restrições para o problema, e, para este exemplo em específico, é a solução ótima.

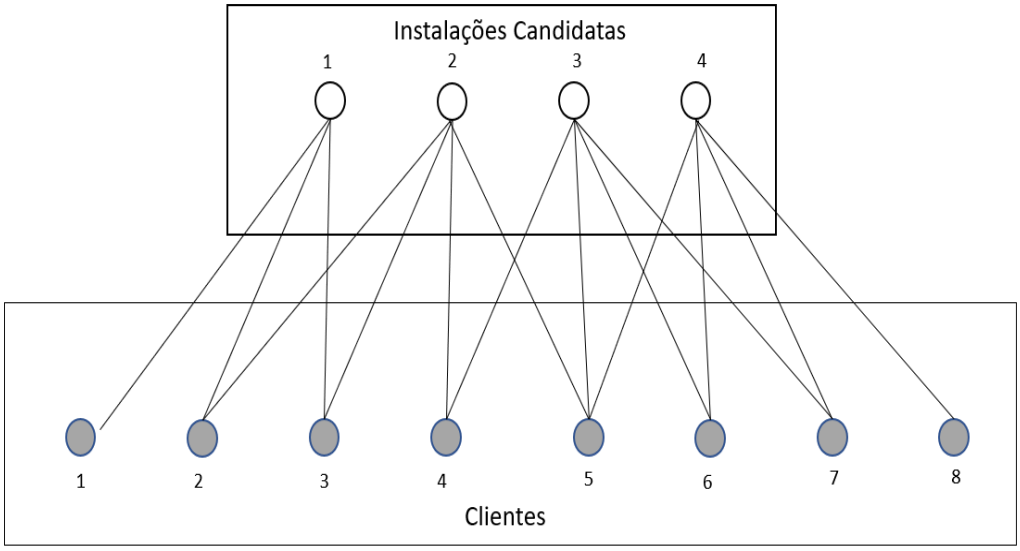

Figura 2.1: Representação gráfica de possíveis relações de coberturas para a instância de exemplo com 4 instalações em potencial e 8 clientes.

Podemos ainda analisar outras possibilidades de cobertura parcial. Considere que, para a mesma demanda solicitada anteriormente, tem-se os seguintes vetores de soluções:

$$\begin{aligned} y &= [0, 1, 0, 0] \\ z &= [0, 1, 1, 1, 1, 0, 0, 0] \end{aligned}$$

Nesta solução, a instalação 2 está aberta, o que acarreta um custo de abertura de 15, e tem-se uma demanda coberta com o valor de 105. Logo, o valor da demanda atende às restrições para o problema de cobertura parcial, porém, o custo de abertura é maior que o da solução encontrada anteriormente. Sendo assim, tem-se uma solução viável, que, porém, não é ótima.

Capítulo 3

Trabalhos Relacionados

Problemas de localização baseados em cobertura fazem parte de uma classe de problemas de localização de grande importância. Estes problemas consistem em determinar a localização de instalações de forma a cobrir demandas de clientes. Neste tipo de problemas, em geral, um cliente é considerado coberto por uma instalação se a distância entre os dois for menor ou igual a uma distância máxima previamente definida, denominada raio de cobertura.

Em Sherali et al. (1991) é apresentada uma aplicação de problemas probabilísticos a problemas de cobertura parcial de conjuntos. Neste trabalho, os autores consideram uma situação em que instalações precisam ser construídas para atender a uma determinada quantidade de clientes. Porém, o conjunto de novas instalações a serem instaladas não consegue garantir uma cobertura absoluta para todos os clientes. Foi formulado, então, um modelo de maximização da confiabilidade total do serviço sujeito a uma restrição orçamentária. Foi implementado e testado um algoritmo de *Branch-and-Bound* e avaliados os efeitos de parâmetros do problema na dificuldade de resolução. Foram identificadas boas soluções para instâncias com 15 instalações e 100 clientes a serem atendidos. Porém, nos testes com instâncias maiores que essas, os autores propuseram uma nova abordagem de resolução para o problema, incluindo a utilização de heurísticas. Além disso, os autores propõem a inclusão de novos níveis de restrições no problema para obtenção de melhores níveis de confiabilidade nas soluções.

Daskin & Owen (1999) propõem dois novos problemas de cobertura, sendo um deles uma abordagem para problema de cobertura de conjuntos parcial para solução do problema de p-centro. No problema de p-centro, são associados, a cada nó de demanda, pesos estritamente positivos, relativos à demanda e à população no nó. O objetivo é minimizar a distância máxima entre um nó e sua instalação mais próxima ponderada pela demanda. Os autores propuseram um algoritmo lagrangeano para resolver o problema de p-centro de cobertura parcial. Como o problema de cobertura

de conjunto parcial, este problema requer o número mínimo de instalações necessárias para que as demandas possam ser cobertas dentro de uma distância definida previamente. O problema foi resolvido fixando-se a distância de cobertura e realizando uma busca bissecional no número de instalações necessárias para garantir que uma demanda mínima seja atendida. O problema foi resolvido para um conjunto de instâncias com 150 nós e com distâncias de cobertura de 200, 300, 400 e 500 milhas, bem como os percentuais de demanda a serem atendidos variando entre 90 a 100%. O trabalho obteve bons resultados e, em todos os casos, os tempos de resolução foram inferiores a 11 segundos

Vasko et al. (2003) apresentam um estudo para o problema de cobertura parcial aplicado ao problema de localização não capacitado. Os autores partem da generalização do problema de localização de instalações não capacitadas, em que busca-se determinar um número mínimo de armazéns a serem abertos, de modo que todas as lojas de varejo sejam atendidas por um depósito e a soma dos custos fixos de entrada e operação dos armazéns e os custos variáveis de abastecimento das lojas de varejo pelos armazéns abertos é minimizado. Nesta abordagem é introduzido um único armazém, descrito como "armazém universal", que pode atender todas as lojas de varejo e possui custo fixo zero. Nesse trabalho é considerado, como solução desejada, o atendimento de um percentual entre 85 a 90% de cobertura. O algoritmo de resolução foi implementado no software DUALOC e foi reportado que o mesmo obteve soluções ótimas até mesmo para conjuntos de instâncias de grandes dimensões.

Em Bilal et al. (2014), é proposto um algoritmo híbrido heurístico para resolver uma nova variante do problema de cobertura parcial de conjuntos, em que o objetivo é maximizar o lucro e não é necessário cobrir todos os elementos. Foi apresentada uma aplicação industrial do modelo e proposto um algoritmo heurístico híbrido para resolvê-lo. O estudo é iniciado após se realizar a análise de exploração e viabilidade de localizações em potenciais de depósitos de minerais. O principal objetivo é selecionar os pontos de perfuração para cobrir a maioria dos locais potenciais com uma quantidade mínima de perfuração. Minimizar a quantidade de perfuração envolve escolher o número, localização, orientação e comprimento de cada furo para minimizar o custo total da perfuração. Além do custo da perfuração, mover a plataforma de perfuração de um local para outro também é caro. Os locais onde a plataforma de perfuração podem ser posicionadas são chamados de estações de perfuração. Para a solução do problema, é proposto um algoritmo heurístico híbrido que combina duas metaheurísticas: *Iterated Local Search* (ILS) e *Tabu Search* (TS). Para a realização dos testes, foram utilizados dados geométricos reais (modelos de blocos e furos) da indústria de mineração. O modelo matemático também foi implementado no software IBM CPLEX para que fosse possível verificar a eficiência dos algoritmos propostos.

Os resultados computacionais demonstraram que o algoritmo obteve êxito na maioria das soluções encontradas, sendo grande parte das soluções ótimas ou muito próximas da ótima.

Cordeau et al. (2019) trazem uma abordagem mais recente para o problema de cobertura parcial de conjuntos. Nesse trabalho é apresentada a aplicação do método de decomposição de Benders aos problemas de cobertura máxima e cobertura parcial, com foco em reduzir o tempo computacional nas soluções de instâncias, em que o número de clientes é muito maior que o número de instalações potenciais. Foi proposto um algoritmo de decomposição de Benders, implementado por meio da linguagem `C++`, que foi comparado ao *solver* CPLEX da IBM. Nos dois casos, o algoritmo demonstrou grande eficiência em comparação ao CPLEX, obtendo soluções ótimas em tempo computacional relativamente pequeno para um conjunto de instâncias com até 40 milhões de clientes e 100 instalações candidatas. Em Cardoso et al. (2022) é proposto um algoritmo fundamentado na metaheurística *Iterated Local Search (ILS)* para solucionar o mesmo conjunto de instâncias utilizados por Cordeau et al. (2019), que obteve resultados sólidos quanto à qualidade das soluções obtidas, bem como aos tempos computacionais, quando comparados ao *solver* CPLEX.

Em Coco et al. (2022) são abordados dois problemas, sendo o primeiro conhecido como Problema de Cobertura de Conjunto Ponderado e o segundo o Problema de Cobertura do Conjunto de Benefícios Máximos. Os autores abordam as contrapartes de otimização robusta de ambos os problemas. Nos dois casos, a incerteza nos dados é modelada usando um intervalo de valores contínuos, representando todos os infinitos valores que cada parâmetro incerto pode assumir. Para solução dos problemas, foram propostos algoritmos exatos baseados em decomposição de Benders e no método de *Branch-and-Cut*. Apesar de ambos os problemas terem como objetivo minimizar o máximo arrependimento, os algoritmos exatos obtiveram desempenho diferentes para cada problema. Os algoritmos aplicados ao Problema de Cobertura de Conjunto Ponderado obtiveram bons resultados, enquanto os algoritmos para o Problema de Cobertura de Conjunto de Benefício Máximo não produziu bons resultados. A diferença se deve ao fato de que ambos os problemas pertencerem a diferentes classes de complexidade.

Um breve resumo dos trabalhos referenciados é apresentado na Tabela 3.1.

Tabela 3.1: Trabalhos Relacionados

Referência	Problema	Abordagem de resolução
Daskin & Owen (1999)	Problema de p-Centro e Cobertura Parcial de Conjuntos	Algoritmo Lagrangeano e busca bissecional
Bilal et al. (2014)	Problema de Cobertura de Conjunto Parcial com maximização de lucro	Heurística ILS e Busca tabu
Vasko et al. (2003)	Problema de Cobertura de Conjunto Parcial	Resolução do problema pelo software DUALOC
Cordeau et al. (2019)	Problema de Cobertura de Conjunto Parcial e Cobertura Máxima	Decomposição de Benders
Cardoso et al. (2022)	Problema de Cobertura de Conjunto Parcial	Meta-Heurística Iterated Local Search (ILS)
Vatsa & Jayaswal (2021)	Problema de Localização de Cobertura Máxima	Otimização Robusta sob Condições de Incerteza
Coco et al. (2022)	Problema de Cobertura de Conjunto Ponderado e de Benefícios Máximos	Otimização Robusta sob Condições de Incerteza
Cardoso et al. (2022)	Problema de Cobertura de Conjunto Parcial	Algoritmo ILS

Capítulo 4

Fundamentação Teórica

Neste capítulo será apresentada a fundamentação teórica utilizada neste trabalho. Inicialmente, na Seção 4.1, será realizada uma introdução acerca dos métodos meta-heurísticos de modo amplo. Posteriormente, na Seção 4.2, é apresentada a descrição de procedimentos de construção gulosa e aleatórios. Em sequência, na Seção 4.3, é descrito o método *Variable Neighborhood Descent* (VND). Após isso, faz-se a descrição da meta-heurística *Iterated Local Search* (ILS) na Seção 4.4. Finalmente, é realizada a apresentação do método *Simulated Annealing* (SL) na Seção 4.5.

4.1 Métodos Meta-heurísticos

Uma heurística pode ser entendida como um método de resolução de problemas que encontra soluções satisfatórias em tempo computacional aceitável, porém, sem a garantia de encontrar uma solução ótima. Muitas definições podem ser dadas sobre o termo meta-heurística. Resumidamente, conforme descreve Ribeiro (1996), pode-se dizer que são procedimentos que guiam outras heurísticas para explorar o espaço de soluções.

Geralmente, conforme apresentado por Osman & Kelly (1997), as meta-heurísticas são divididas em duas etapas. A primeira fase é a fase construtiva, que consiste na construção de uma solução inicial, elemento por elemento, sucessivamente, até que se tenha uma solução completa. Posteriormente, a fase de refinamento da solução é responsável por melhorar a solução inicial.

Neste trabalho, para a construção da solução inicial, foi utilizada uma heurística construtiva randômica e, para refinamento desta solução, foram utilizados um algoritmo que combina a busca local *Variable Neighborhood Descent* com a meta-heurística *Iterated Local Search* (ILS) e um algoritmo baseado na meta-heurística *Simulated Annealing* (SA).

4.2 Heurística Construtiva

Dentre os principais procedimentos de construção de uma solução, tem-se heurísticas construtivas gulosas, que serão apresentadas na Seção 4.2.1, e heurísticas construtivas aleatórias, que serão apresentadas na Seção 4.2.2.

4.2.1 Construção Gulosa

O método de construção gulosa consiste em construir uma solução inicial a partir de inserção iterativa de novos elementos a uma solução parcial que fornece o maior benefício para a função objetivo, de acordo com um determinado critério. Este método consiste em selecionar um candidato, em um conjunto de candidatos disponíveis, e inseri-lo na solução parcial corrente. Em seguida, é calculado qual é o próximo candidato que fornece o maior benefício para a função objetivo para compor a solução. Novos candidatos são inseridos na solução sucessivamente até que a solução esteja completa.

Importante reforçar que este procedimento guloso mencionado, também é conhecido como método de inserção mais barata, no qual os candidatos que possuem o melhor benefício são escolhidos individualmente e inseridos na função objetivo, de modo a se obter o melhor benefício para a solução.

O procedimento para a construção gulosa é apresentado no Algoritmo 1.

Algoritmo 1 Procedimento ConstrucaoGulosa $(g(.), s)$

```
1:  s ← ∅
2:  Inicialize o conjunto C de elementos candidatos
3:  while C ≠ ∅ do
4:      g_melhor ← melhor{g(t)|t ∈ C}
5:      t_melhor ← o valor de t ∈ C associado a g_melhor
6:      if t_melhor pode ser inserido na solução parcial then
7:          s ← s ∪ {t_melhor}
8:      end if
9:      C ← C \ {t_melhor}
10: end while
11: Retorne s
12: fim Construção Gulosa
```

O algoritmo se inicia na linha 1 com uma solução vazia; na linha 2, tem-se o conjunto de todos os elementos candidatos. Inicia-se, então, na linha 3, uma estrutura de repetição, que percorre as soluções candidatas C. Entre as linhas 4 e 7, é verificado se o valor de t, que pertence ao conjunto C e que possui o melhor valor de função de avaliação, denominado t_{melhor}, pode ser inserido na solução parcial. Caso t_{melhor}

possa ser inserido na solução parcial, a variável s é atualizada. O procedimento se encerra após percorrer todas os elementos candidatos e é retornado o melhor valor para s, que será a solução inicial construída.

4.2.2 Construção Aleatória

O método de construção aleatória de solução inicial, apresentado no Algoritmo 2, consiste em construir uma solução elemento por elemento, em que um único elemento candidato, selecionado aleatoriamente dentre os elementos disponíveis, é adicionado à solução parcial a cada passo. Esse processo continua até que se tenha uma solução inicial para o problema ou até que seja atingido um determinado limite. Este procedimento pode ter a vantagem, em relação ao procedimento de construção gulosa, de demandar menor esforço computacional, uma vez que não requer a avaliação de todos os candidatos a cada iteração do algoritmo. Por outro lado, por não considerar o critério de otimização na construção, as soluções iniciais obtidas podem não ser de boa qualidade.

Algoritmo 2 Procedimento ConstrucaoAleatória(s)

```
1: s ← ∅
2: Inicialize o conjunto C de elementos candidatos
3: while C ≠ ∅ do
4:     t_alea ← o valor de t ∈ C gerado aleatoriamente
5:     if t_alea pode ser inserido na solução parcial then
6:         s ← s ∪ {t_alea}
7:     end if
8:     C ← C \ {t_alea}
9: end while
10: Retorne s
11: fim Construção Aleatória
```

4.3 Heurística de Refinamento VND

O método de busca local *Variable Neighborhood Descent*(VND), proposto por Mladenović & Hansen (1997), é um procedimento de busca local que explora o espaço de soluções em vizinhanças gradativamente mais distantes. O método VND é um método de descida em vizinhança variável, que explora o espaço de soluções por trocas sistemáticas de estruturas de vizinhança.

Resumidamente, o algoritmo segue os seguintes princípios básicos:

- Um ótimo local com relação a uma vizinhança não corresponde necessariamente a um ótimo com relação a outra vizinhança;

- Um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma vizinhança são relativamente próximos.

No método VND básico, proposto em Mladenović & Hansen (1997), sempre que há uma melhora em uma certa vizinhança, retorna-se à estrutura de vizinhança inicial. O procedimento básico proposto para o método VND é descrito no Algoritmo 3

Algoritmo 3 VND básico

```
1:  Seja s_0 uma solução inicial e r o número de estruturas de vizinhança
2:  s ← s_0
3:  k ← 1
4:  while k ≤ r do
5:      Encontre o melhor vizinho s' ∈ N^(k)(s)
6:      if (f(s') ≤ f(s)) then
7:          s ← s'
8:          k ← 1
9:      else
10:         k ← k + 1
11:     end if
12: end while
13: Retorne s
```

O algoritmo parte de uma solução inicial s_0 e considera r estruturas de vizinhanças ordenadas, em que a estrutura de vizinhança k é denotada por N^k. Assim, em cada iteração, é feita uma busca local na solução corrente. Caso seja encontrada uma solução melhor, a variável s é atualizada e a busca continua nesta estrutura de vizinhança. Caso contrário, finaliza-se a busca nesta estrutura de vizinhança e se inicia uma nova busca na estrutura de vizinhança seguinte. Ao final da execução, o algoritmo encerra, retornando uma solução que é um ótimo local com relação a todas estruturas de vizinhança exploradas.

4.4 *Iterated Local Search* (ILS)

A meta-heurística *Iterated Local Search* (ILS), proposta por Lourenço et al. (2003), apresenta uma estratégia, usando perturbações, para que a busca não fique presa em ótimos locais. No método ILS, aplica-se perturbações na solução ótima local corrente e, em seguida, aplica-se um método de busca local a esta solução. Esta perturbação não pode ser fraca, pois neste caso a solução poderá não sair do ótimo

local encontrado, e também não deve ser muito forte, evitando assim um reinício aleatório.

O princípio do método se baseia no seguinte. Para melhorar o valor da função objetivo e escapar de um ótimo local, é necessário reiniciar a busca local a partir de um outro ponto de partida. Sendo assim, uma possível alternativa seria selecionar um novo ponto de partida de modo aleatório. Porém, essa abordagem tende a ser computacionalmente ineficiente, de modo que o conjunto de soluções exploradas aumenta significativamente, de acordo com o tamanho da instância utilizada. Em contrapartida, ao realizar somente uma busca local sem uma perturbação, não será possível explorar o espaço de busca e, assim, a busca ficaria presa em um ótimo local. Buscando atender estes dois aspectos surge o método ILS, sendo um procedimento que combina as duas ideias, ou seja, o reinício em um novo ponto e a aplicação de busca local para a determinação de uma nova solução.

Conforme dito anteriormente, para conseguir uma melhor exploração do espaço de busca e não ser completamente aleatório, o método ILS explora as vizinhanças e gera perturbações que permitem a busca atingir vizinhanças intermediárias.

Segundo Lourenço et al. (2003), basicamente o método funciona da seguinte maneira: a primeira etapa do algoritmo parte de uma solução inicial, na qual será realizada uma busca local. Após isso, é feita uma perturbação na solução corrente, seguida de uma busca local na solução perturbada. A solução corrente pode ser atualizada ou não pela nova solução obtida, a partir da avaliação de um dado critério de aceitação. Caso a nova solução não seja aceita, o nível de perturbação é aumentado e o procedimento é realizado novamente, até que essa perturbação atinja um determinado nível. Este procedimento de busca local conduz a uma boa exploração do espaço de busca, desde que as perturbações não sejam muito pequenas, nem grandes demais. Se forem muito pequenas, poucas novas soluções serão exploradas. Se, ao contrário, as perturbações forem muito grandes, o procedimento será semelhante a um algoritmo do tipo de reinicialização aleatória (Lourenço et al., 2003). Uma ilustração do comportamento do procedimento ILS global é mostrada na Figura 4.1.

Pode-se então sintetizar o método em quatro componentes:

(i) Geração de uma solução inicial;

(ii) Busca Local: utilizada para retornar uma solução melhor;

(iii) Perturbação: utilizada para alterar a solução encontrada pela busca local (solução corrente);

(iv) Critério de aceitação: utilizado para decidir em qual solução a próxima busca local será aplicada.

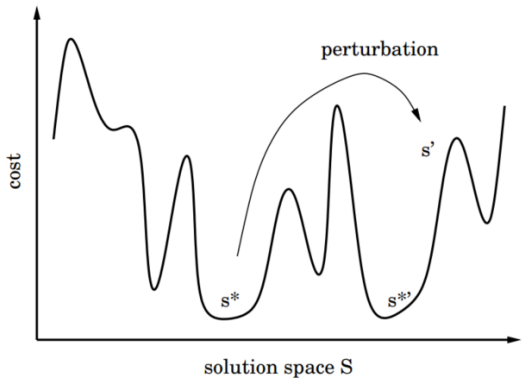

Figura 4.1: Procedimento de exploração do espaço de busca por meio de perturbações. Fonte: Lourenço et al. (2003)

O pseudo-código do ILS é apresentado no Algoritmo 4. Inicialmente, é gerada uma solução inicial, que é atribuída à variável s_0, conforme apresentado na linha 1. Em seguida, faz-se uma busca local na solução s_0, atribuindo-se à solução melhorada a variável s, sendo esta, então, a solução corrente. Logo após, são inicializadas as variáveis $iter$, utilizada para realizar a contagem do número de iterações, e $MelhorIter$, que contabiliza o número de iterações sem melhora. Inicia-se, então, o laço de repetição condicional, no qual é realizada uma perturbação na solução corrente, conforme será detalhado logo mais. Posteriormente, é realizada a busca local na solução perturbada s' encontrada, dando origem a uma solução s''. Em seguida, é verificado o critério de aceitação. Se a solução s'' for melhor que a solução corrente s, a solução s é atualizada para s'' e o número de iterações sem melhora é zerado. Caso contrário, a solução s continua sendo igual à solução corrente e o número de iterações sem melhora é incrementado. Este procedimento é repetido enquanto o número de iterações sem melhora for menor ou igual a $ILS_{\max}$, que é um parâmetro do algoritmo.

Algoritmo 4 *ILS*

1: $s_0 \leftarrow SolucaoInicial()$
2: $s \leftarrow BuscaLocal(s_0)$
3: $Iter \leftarrow 0$;
4: $MelhorIter \leftarrow Iter$; {Iteração em que ocorreu melhora}
5: **while** $iter - MelhorIter \leq ILS_{\max}$ **do**
6: $\quad iter \leftarrow iter + 1$
7: $\quad s' \leftarrow perturbacao(s, nivel)$
8: $\quad s'' \leftarrow BuscaLocal(s')$
9: $\quad (s, MelhorIter) \leftarrow CriteriosAceitacao(s, s'')$
10: **end while**
11: Retorne s

4.5 *Simulated Annealing* (SA)

A meta-heurística *Simulated Annealing* (SA), proposta por Kirkpatrick et al. (1983), é um procedimento que faz analogia ao processo metalúrgico de recozimento de metais para a obtenção de estados de baixa energia de um sólido. Neste caso, os possíveis estados físicos da temperatura dos metais correspondem a soluções no espaço de busca e a energia de cada estado é correspondente ao valor da função objetivo. O método SA utiliza a fundamentação teórica do Algoritmo de Metropolis et al. (1953) através de um esquema de redução gradual de temperatura. Para este processo, o resfriamento lento da temperatura é essencial para a manutenção do equilíbrio térmico, pois, caso ocorra um resfriamento abrupto, o material poderá ficar com sua resistência física frágil.

Outro fator relevante do método SA é a possibilidade de aceitação de soluções piores que a solução corrente, sendo possível, dessa forma, não ficar presos em mínimos locais. Deste modo, temos a possibilidade de que o algoritmo busque soluções melhores de forma mais eficiente em ambientes que podem ter diversos mínimos locais. Conforme descrito por Goffe et al. (1994), o método pode ser considerado uma técnica de busca estocástica que usa um processo de perturbação aleatória para explorar diferentes regiões do espaço de busca. Este processo de perturbação randômica é controlado pela função de temperatura, que é reduzida gradualmente ao longo das iterações, diminuindo assim a taxa de aceitação de soluções piores no decorrer da busca.

4.5.1 Fundamentação do Método do *Simulated Annealing*

Em um processo metalúrgico, após a etapa de laminação a quente, o material é submetido ao processo de recozimento. Nesse processo, o metal é aquecido a uma temperatura elevada e resfriado de forma lenta. Durante o recozimento, o material passa por uma série de estados possíveis em um período de tempo suficientemente longo e, nesta fase, os átomos da estrutura percorrem todos os estados acessíveis e se acomodam na posição de menor energia interna.

Quando o resfriamento é realizado de maneira suficientemente lenta, obtém-se uma estrutura cristalina livre de imperfeições, caracterizada por um estado de baixa energia. Esse resfriamento lento resulta em produtos mais estáveis, estruturalmente fortes e com menor energia interna. Por outro lado, quando o resfriamento é rápido, ocorre a formação de produtos metaestáveis, que possuem maior energia interna. Esses produtos não alcançam o mesmo nível de estabilidade e resistência estrutural observados no caso do resfriamento lento. Assim, o controle adequado do processo de resfriamento no recozimento é essencial para obter produtos com propriedades

desejáveis, sendo o resfriamento lento preferível para alcançar uma estrutura cristalina estável e energeticamente favorável.

Conforme descrito por Goffe et al. (1994), o método pode ser considerado uma técnica de busca estocástica, sendo possível realizar uma analogia com um problema combinatório, uma vez que os estados possíveis de um metal correspondem a soluções do espaço de busca, a energia em cada estado corresponde ao valor da função objetivo e a energia mínima (se o problema for de minimização ou máxima, se de maximização) corresponde ao valor de uma solução ótima local, possivelmente global.

4.5.2 Algoritmo de Metrópolis

O *Simulated Annealing* aplica o Algoritmo de Metropolis et al. (1953), durante o processo iterativo. Assim, a cada iteração, um novo estado é gerado a partir do estado corrente através de uma modificação aleatória. Em um problema de minimização, se o novo estado apresenta uma energia menor que o estado corrente, o novo estado é adotado como estado corrente para a próxima iteração.

No entanto, se o novo estado possui uma energia maior que o estado corrente, a decisão de mudar do estado corrente para o novo estado é determinada de forma probabilística. Neste caso, uma probabilidade é calculada levando em consideração fatores como a diferença de energia entre os dois estados e a temperatura atual do sistema. Quanto maior a diferença de energia e menor a temperatura, menor será a probabilidade de se mudar para o novo estado. Isso implica que, à medida que a temperatura diminui ao longo das iterações, a chance de se aceitar um estado de energia maior diminui, o que favorece a busca por estados de menor energia e, consequentemente, pela solução ótima do problema de minimização.

Portanto, o processo de aceitar ou rejeitar um novo estado com energia maior que o estado corrente é determinado por uma probabilidade, influenciada pela diferença de energia e pela temperatura atual do sistema. Essa abordagem probabilística permite explorar o espaço de soluções em busca de uma solução ótima ou próxima dela. Este procedimento é baseado no cálculo de probabilidade de aceitação de um estado com maior energia com base na função:

$$e^{-\frac{\triangle}{\kappa T}}, \tag{4.1}$$

em que κ igual a constante de Boltzmann, T é a temperatura corrente e $\triangle$ é a diferença de energia entre o novo estado e o estado corrente.

4.5.3 Algoritmo *Simulated Annealing*

O Algoritmo 5 apresenta o método *Simulated Annealing* aplicado a um problema de minimização.

Algoritmo 5 SimulatedAnnealing $(f(.), N(.), \alpha, SAmax, T_0, s)$

```
1: s* ← s            Melhor solução corrente
2: IterT ← 0         Número de iterações na temperatura T
3: T ← T_0           Temperatura corrente
4: while T > 0 do
5:     while IterT < SAmax do
6:         IterT ← IterT + 1
7:         Gere uma nova solução s' ∈ N(,)
8:         Δ = f(s') − f(s)
9:         if (Δ < 0) then
10:            s ← s'
11:            if (f(s') < f(s*)) then
12:                s* ← s'
13:            end if
14:        else x ∈ U(0,1)
15:            if (x < e^{−Δ/T}) then
16:                s ← s'
17:            end if
18:        end if
19:    end while
20:    T ← α × T
21:    IterT ← 0
22: end while
23: Retorne s*
```

O algoritmo inicia o processo de otimização partindo de uma solução inicial s, que é atribuída como a solução corrente. Com o valor da temperatura T inicializado pela temperatura inicial T_0, inicia-se a busca por novas soluções de modo aleatório. A cada nova iteração do método, um novo vizinho s' na estrutura de vizinhança $\mathcal{N}$ é gerado. Em seguida, é avaliada a variação da função objetivo $\Delta f = f(s') - f(s)$, podendo ocorrer as seguintes situações:

- $\Delta f < 0$: houve redução no nível de energia, significando que a solução atual encontrada é melhor que a solução anterior. Logo, essa solução é aceita como nova solução corrente;
- $\Delta f = 0$: não houve variação do estado de energia e essa solução s' será aceita como solução corrente;
- $\Delta f > 0$: houve um aumento no estado de energia. Assim, se faz necessário

a verificação probabilística da solução para verificação do seu aceite ou não. Esta verificação é baseada na Equação 4.1, em que há maior probabilidade das soluções serem aceitas em temperaturas mais elevadas que em temperaturas mais baixas.

A temperatura alta no início do procedimento pode possibilitar uma chance maior do algoritmo não ficar preso em mínimos locais, pois são aceitas mais soluções avaliadas, conforme critério de avaliação probabilístico apresentado na linha 15, em que a nova solução será aceita se tivermos um valor maior que um número entre zero e um, gerado aleatoriamente. Caso contrário, a solução atual é mantida. No entanto, à medida que a temperatura se aproxima de zero, o algoritmo se comporta como um algoritmo de descida, uma vez que a probabilidade de aceitação de movimentos que tendem a piorar a solução diminui. O método é finalizado quando a temperatura se aproxima de zero. Neste momento do procedimento as soluções que pioram o valor da função objetivo não são mais aceitas, o que significa que o sistema se estabilizou e evidencia o encontro de um ótimo local.

4.5.4 Temperatura Inicial

Neste trabalho, foi utilizado um procedimento baseado em simulação para determinar a temperatura inicial, conforme utilizado por Júnior et al. (2005). Este procedimento é apresentado no Algoritmo 6, que considera que uma temperatura inicial ideal é aquela que possui uma taxa de aceitação, denotada por γ, alta, isto é, próxima a 1. Partindo de uma temperatura inicial T_0 baixa qualquer, é efetuado $SAmax$ iterações do *Simulate Annealing* nessa temperatura. Ao final dessas $SAmax$ iterações, se a quantidade de soluções aceitas for maior ou igual a $\gamma \times SAmax$ movimentos, então podemos garantir que temos uma temperatura alta o suficiente para o início do procedimento do *Simulate Annealing*; caso contrário, devemos aumentar a temperatura inicial do procedimento, o que é realizado usando o parâmetro β.

Outro tipo de abordagem que pode ser usado para determinar a temperatura inicial é baseada em começar com uma solução específica e gerar todos os seus possíveis vizinhos, ou uma fração desses vizinhos. Para cada um desses vizinhos, o custo correspondente é calculado, usando a função de avaliação adequada. Esse procedimento é repetido para outros pontos iniciais, uma vez que o custo das soluções vizinhas pode variar, dependendo da solução inicial adotada. A maior dentre essas estimativas de custo é, então, considerada como uma estimativa para a temperatura inicial.

Em princípio, a temperatura final ideal deveria ser zero. No entanto, na prática, é suficiente chegar a uma temperatura próxima de zero, devido às limitações de precisão da implementação computacional utilizada. Essa aproximação considera

Algoritmo 6 TemperaturaInicial $(\beta, \gamma, SAmax, T_0, s)$

1: $T \leftarrow T_0$
2: Continua $\leftarrow$ TRUE
3: **while** Continua **do**
4: Aceitos $\leftarrow 0$ *Soluções aceitas na temperatura T*
5: **for** $Iter = 1$ to $SAmax$ **do**
6: $\Delta = f(s') - f(s)$
7: **if** $\Delta < 0$ **then**
8: Aceitos $\leftarrow$ Aceitos $+1$
9: **else** Gere $x \in U(0, 1)$
10: **if** $(x < e^{-\Delta/T})$ **then**
11: Aceitos $\leftarrow$ Aceitos $+1$
12: **end if**
13: **end if**
14: **end for**
15: **if** Aceitos $\geq \gamma \times SAmax$ **then**
16: Continua $\leftarrow$ FALSE
17: **else** $T \leftarrow \beta \times T$
18: **end if**
19: **end while**
20: Retorne T

que a precisão dos cálculos e a resolução das variáveis do problema não permitem uma temperatura absolutamente zero. Portanto, é aceitável alcançar uma temperatura próxima de zero, levando em conta as limitações inerentes à implementação computacional utilizada.

Capítulo 5

Metodologia

Este capítulo apresenta os métodos utilizados para resolução de instâncias do PS-CLP, incluindo a representação da solução, as estruturas de vizinhanças exploradas, bem como os algoritmos utilizados. A Seção 5.1 mostra a representação da solução para o problema proposto nesta dissertação. Na Seção 5.2 é feita a apresentação da função de avaliação. Na Seção 5.3 são apresentados algoritmos para a construção da solução inicial. A Seção 5.4 faz a apresentação de uma heurística de refinamento para o problema e, por fim, na Seção 5.5 é feita a apresentação das meta-heurísticas desenvolvidas para a solução do problema objeto de pesquisa neste trabalho.

5.1 Representação da solução

A representação de uma solução para o problema é feita por dois vetores, com papel semelhante ao das variáveis de decisão do modelo matemático, denotados também por y e z. O vetor y é um vetor binário de dimensão n (número de instalações candidatas), em que o valor 1 em uma dada posição i representa que a instalação i está aberta e o valor 0 representa que a instalação i está fechada. Um exemplo para o vetor y para uma solução de um problema com $n = 6$ instalações candidatas e 3 instalações abertas é dado por:

$$y = [1, 1, 0, 0, 1, 0].$$

As posições 1, 2 e 5 do vetor representam as instalações abertas e as posições 3, 4 e 6 representam as instalações fechadas.

O vetor z representa os clientes cobertos, em que o valor 1 em uma dada posição j representa que o cliente j é coberto por uma instalação ativa, e o valor 0, caso contrário. Uma ilustração do vetor z para uma solução em que $m = 10$ pontos de

demanda seria o vetor:

$$z = [1, 1, 0, 0, 1, 0, 1, 1, 0, 0], \tag{5.1}$$

em que apenas os clientes 1, 2, 5, 7 e 8 são cobertos.

5.2 Função de Avaliação

A função de avaliação é representada pela própria função objetivo do problema. Seja $s = (y, z)$ uma solução com vetores de representação da solução y e z, como descritos na Seção 5.1. Então, o valor da função de avaliação para a solução s, denotado por $f(s)$, é dado por:

$$f(s) = \sum_{i=1}^{n} f_i y_i. \tag{5.2}$$

5.3 Construção da Solução Inicial

Neste trabalho foi utilizado o método de construção aleatória para ambos os métodos *Iterated Local Search (ILS)* e *Simulated Annealing (SA)*.

5.3.1 Geração via construção aleatória

A estratégia de construção aleatória é apresentada no Algoritmo 7. Neste algoritmo, a notação $\bar{0}$ representa um vetor nulo com as dimensões adequadas aos vetores y e z. O algoritmo se inicia com todas as instalações fechadas e com nenhum ponto de demanda coberto. Então, no laço principal do algoritmo, seleciona-se aleatoriamente uma instalação fechada para ser aberta. Após esta abertura, assinala-se, de acordo com o raio de cobertura, como cobertos todos os clientes que podem ser cobertos por esta nova instalação. Como cada cliente está associado a uma demanda a ser atendida, as demandas dos clientes cobertos definem a demanda total atendida pela solução corrente. Caso a demanda total atendida pela solução corrente seja maior ou igual a D, o procedimento é encerrado, pois se tem uma solução factível. Caso contrário, é realizada uma nova ativação de modo aleatório de uma nova instalação e repete-se novamente todo o procedimento, até que o critério de parada referente ao atendimento da demanda seja satisfeito. Ao finalizar este procedimento, a solução construída é retornada.

Algoritmo 7 ConstrucaoAleatoria ()

1: $y \leftarrow \bar{0}$
2: $z \leftarrow \bar{0}$
3: $\bar{d} \leftarrow 0$
4: $C \leftarrow$ Conjunto de todas as instalações candidatas
5: **while** $\bar{d} < D$ **do**
6: $\quad i \leftarrow$ Seleciona aleatoriamente uma instalação em C
7: $\quad C \leftarrow C \setminus \{i\}$
8: $\quad y_i \leftarrow 1$
9: $\quad z_j \leftarrow 1$, para todo j tal que $a_{ij} = 1$
10: $\quad \bar{d} \leftarrow$ demanda total coberta pelas instalações abertas
11: **end while**
12: Retorne $s_0 = (y, z)$
13: **fim** ConstrucaoAleatoria

5.4 Heurística de Refinamento

Para explorar o espaço de soluções na resolução do algoritmo *Iterated Local Search (ILS)*, foi utilizado o método de Descida em Vizinhança Variável (*Variable Neighborhood Descent* – VND), proposto por Mladenović & Hansen (1997). O método VND explora o espaço de soluções por meio de trocas sistemáticas de estruturas de vizinhança. Para o problema proposto, foram utilizadas duas estruturas de vizinhança, baseadas em dois movimentos.

O primeiro movimento é o movimento de trocas de bits, em que ocorre a abertura ou o fechamento de instalações. Este movimento tem a vantagem de fornecer a garantia de exploração de todo o espaço de soluções para o problema. O segundo movimento é o de troca de instalações, que fecha uma instalação aberta e abre uma instalação fechada. Porém, com esta estrutura de vizinhança, não se tem a garantia da exploração de todo o espaço solução, já que a quantidade de instalações abertas é fixa. No entanto, esta estrutura de vizinhança é necessária, pois foi observado que, em algumas situações, não é possível encontrar uma solução factível melhor apenas com o fechamento ou abertura de uma única instalação.

O Algoritmo 8 descreve a implementação realizada do método VND. O algoritmo parte de uma solução inicial s_0 e considera r estruturas de vizinhanças. Neste caso, o valor de r será igual a 2. As estruturas de vizinhanças são denotadas por N^k para $k = 1, 2$. As estruturas mencionados foram ordenadas da seguinte forma:

(i) troca de bits; seguida por

(ii) troca de instalações.

Assim, em cada iteração, é feita uma busca local na solução corrente através da

primeira estrutura de vizinhança, ou seja, vizinhança baseada na troca de bits. Caso seja encontrada uma solução melhor, a variável s é atualizada e a busca continua nesta estrutura de vizinhança. Caso contrário, finaliza-se a busca nesta estrutura de vizinhança e se inicia uma nova busca na estrutura de vizinhança referente à troca de instalações. Nesta etapa, caso ocorra melhoria na solução corrente s, esta solução é atualizada e a busca retorna para a primeira estrutura de vizinhança. Caso contrário, o algoritmo se encerra, retornando uma solução que é um ótimo local com relação às duas estruturas de vizinhança. A exploração das duas estruturas de vizinhanças é baseada na estratégia de melhor vizinho, em que a vizinhança completa da solução corrente é explorada.

Algoritmo 8 VND - *Variable Neighborhood Descent*

1: Seja s_0 uma solução inicial
2: $s \leftarrow s_0$
3: $k \leftarrow 1$
4: **while** $k \leq 2$ **do**
5: Encontre o melhor vizinho $s' \in N^{(k)}(S)$
6: **if** $(f(s') \leq f(s))$ **then**
7: $s \leftarrow s'$
8: $k \leftarrow 1$
9: **else**
10: $k \leftarrow k + 1$
11: **end if**
12: **end while**
13: Retorne s

5.5 Meta-heurísticas

Esta seção descreve as meta-heurísticas desenvolvidas para a solução do problema objeto de interesse nessa dissertação. A Seção 5.5.1 apresenta o algoritmo ILS implementado e a Seção 5.5.2 descreve o algoritmo SA.

5.5.1 Algoritmo ILS

Como mencionado no Capítulo 4, o procedimento ILS (Lourenço et al., 2003) é um método que realiza busca local em novas soluções geradas por perturbações na solução ótima local corrente. A meta-heurística *Iterated Local Search* (ILS), descrita no Algoritmo 9, é adotada para a solução do problema.

Inicialmente, é gerada uma solução inicial, que é atribuída à variável s_0, conforme apresentado na linha 1 do Algoritmo 9. Em seguida, faz-se uma busca local na solução

s_0, atribuindo-se a solução melhorada à variável s, sendo esta então a solução corrente. Logo após, são inicializadas as variáveis *iter*, utilizada para realizar a contagem do número de iterações, e h, que contabiliza o número de iterações sem melhora. Inicia-se, então, o laço de repetição condicional, no qual é realizada uma perturbação na solução corrente, conforme será detalhado logo mais. Posteriormente, é realizada a busca local na solução perturbada s' encontrada, dando origem a uma solução s''. Em seguida, é verificado o critério de aceitação. Se a solução s'' for melhor que a solução corrente s, a solução s é atualizada para s'' e o número de iterações sem melhora é zerado. Caso contrário, a solução s continua sendo igual à solução corrente e o número de iterações sem melhora é incrementado. Este procedimento é repetido enquanto o número de iterações sem melhora for menor ou igual a $ILS_{\max}$, que é um parâmetro do algoritmo.

O procedimento de perturbação é apresentado no Algoritmo 10 e consiste na aplicação de uma quantidade de movimentos aleatórios de troca de bits. O parâmetro *nivel* define a quantidade de movimentos realizada.

Algoritmo 9 $ILS(ILS_{\max}, nivel)$

1: $s_0 \leftarrow ConstrucaoGulosa()$
2: $s \leftarrow BuscaLocal(s_0)$
3: $Iter \leftarrow 0$;
4: $h \leftarrow 0$
5: **while** $h \leq ILS_{\max}$ **do**
6: $\quad iter \leftarrow iter + 1$
7: $\quad s' \leftarrow perturbacao(s, nivel)$
8: $\quad s'' \leftarrow BuscaLocal(s')$
9: $\quad (s, h) \leftarrow CriteriosAceitacao(s, s'')$
10: **end while**
11: Retorne s

Algoritmo 10 Procedimento de Perturbação

1: **perturbação** (s, nível)
2: $s' \leftarrow s$
3: $nModificacoes \leftarrow nivel + 1$
4: $cont \leftarrow 1$
5: **while** $(cont < nModificacoes)$ **do**
6: $\quad s' \leftarrow$ Realiza um movimento aleatório de troca de bits em s'
7: $\quad cont \leftarrow cont + 1$
8: **end while**
9: retorne s'

5.5.2 *Simulated Annealing*

O procedimento SA usado nesta dissertação é apresentado no Algoritmo 11. Como mostrado no Capítulo 4, este algoritmo possui, como parâmetro, a temperatura inicial, que será denotada por T_0. Além disso, após um número de iterações em uma dada temperatura T, número esse denotado por $SAmax$, a temperatura corrente será reduzida de acordo com um fator de resfriamento α. Neste trabalho é usado um fator de resfriamento geométrico, em que α mais próximo de 1 indica um resfriamento mais lento, sendo este essencial para a garantia da manutenção do equilíbrio térmico. Já valores de α próximos de zero indicam um resfriamento abrupto e, por consequência, pode provocar uma rápida convergência para um mínimo local.

O algoritmo inicia o processo de otimização partindo de uma solução inicial s, construída a partir do Algoritmo 7, que é atribuída como a solução corrente. Com o valor da temperatura T inicializado pela temperatura inicial T_0, inicia-se a busca por novas soluções de modo aleatório. A cada nova iteração do método, um novo vizinho s' na estrutura de vizinhança $\mathcal{N}$ é gerado. O algoritmo SA implementado utiliza a estrutura de vizinhança troca de bits.

Em seguida é avaliada a variação da função objetivo $\Delta f = f(s') - f(s)$. Caso $\Delta f \leq 0$, a solução s' é aceita. Caso contrário, é feita uma avaliação conforme critério de avaliação probabilístico apresentado na linha 15.

A temperatura alta no início do procedimento possibilita uma chance maior do algoritmo não ficar preso em mínimos locais, pois são aceitas mais soluções avaliadas conforme o critério de avaliação probabilístico, em que a nova solução será aceita se tivermos um valor maior que um número entre zero e um, gerado aleatoriamente. Caso contrário, a solução atual é mantida. No entanto, à medida que a temperatura se aproxima de zero, o algoritmo se comporta como um algoritmo de descida, uma vez que a probabilidade de aceitação de movimentos que tendem a piorar a solução diminui. O método é finalizado quando a temperatura se aproxima de zero. Neste momento do procedimento, as soluções que pioram o valor da função objetivo não são mais aceitas, o que significa que o sistema se estabilizou e evidencia o encontro de um ótimo local.

Neste trabalho, os valores dos parâmetros $SAmax$ e α foram determinados a partir de um *software* para calibração de parâmetros. A determinação do valor de temperatura inicial é feita através do algoritmo de simulação, apresentado no Algoritmo 6.

Algoritmo 11 SimulatedAnnealing $(f(.), \mathcal{N}(.), \alpha, SAmax, T_0, s)$

1: $s^* \leftarrow s$ *Melhor solução corrente*
2: $IterT \leftarrow 0$ *Número de iterações na temperatura T*
3: $T \leftarrow T_0$ *Temperatura corrente*
4: **while** $T > 0$ **do**
5: **while** $IterT < SAmax$ **do**
6: $IterT \leftarrow IterT + 1$
7: Gere uma nova solução $s' \in \mathcal{N}(s)$
8: $\Delta = f(s') - f(s)$
9: **if** $(\Delta < 0)$ **then**
10: $s \leftarrow s'$
11: **if** $(f(s') < f(s^*))$ **then**
12: $s^* \leftarrow s'$
13: **end if**
14: **else** $x \in U(0, 1)$
15: **if** $(x < e^{-\Delta/T})$ **then**
16: $s \leftarrow s'$
17: **end if**
18: **end if**
19: **end while**
20: $T \leftarrow \alpha \times T$
21: $IterT \leftarrow 0$
22: **end while**
23: Retorne s^*

Capítulo 6

Resultados Computacionais

Neste capítulo serão descritos os resultados alcançados pelos algoritmos desenvolvidos. São apresentadas a descrição das instâncias utilizadas e as configurações dos parâmetros e de *hardware* utilizados na execução de cada algoritmo na Seção 6.1 e as metodologias de análise utilizadas na Seção 6.2. Nas Seção 6.3 e Seção 6.4 são apresentados os resultados obtidos para cada algoritmo e uma validação estatística destes resultados. Na Seção 6.5 é apresentada uma comparação entre os resultados obtidos pelas meta-heurísticas e o CPLEX e ainda uma comparação entre os dois métodos implementados. Por fim na Seção 6.6 serão feitas considerações sobre os resultados obtidos.

6.1 Configuração dos Experimentos Computacionais

Para realização dos experimentos foram utilizados os mesmos conjuntos de instâncias empregados no trabalho de Cordeau et al. (2019). Em todos os testes foram utilizados o número de instalações candidatas igual a 100 e número de clientes igual a 10.000, 50.000 e 100.000. Foram variados o raio de cobertura e o percentual de demanda a ser atendida.

Todos os experimentos computacionais foram implementados na linguagem de programação `C++` utilizando um computador com sistema operacional Windows 10 Pro, processador Intel Core I5 9^a geração 8250U CPU@1.60GHz e 8GB de memória RAM. Como forma de comparar os resultados obtidos pelos algoritmos propostos, o modelo matemático dado pelas expressões (2.9)-(2.13) foi implementado usando a linguagem Concert Technology em `C++` e resolvido através do *solver* IBM ILOG CPLEX 20.1.

Os resultados obtidos foram analisados estatisticamente, sendo essa uma das etapas mais importantes do trabalho. Conforme descrito em Montgomery (2017), a

abordagem estatística do projeto experimental é necessária se desejarmos tirar conclusões significativas dos dados obtidos.

6.2 Método de análise

A fim de avaliar os resultados obtidos pelos algoritmos desenvolvidos, foi realizada uma comparação utilizando o melhor resultado de cada execução. Considerando que esses algoritmos são estocásticos, foi necessário verificar a normalidade das execuções para cada instância, de modo a validar essa metodologia de análise. Posteriormente, foi realizada uma comparação entre o melhor resultado de cada instância e o conjunto de resultados dessa instância, com o objetivo de determinar se há uma variação significativa entre o conjunto de resultados. Caso essa comparação indique que não há variação significativa entre o melhor resultado de cada instância e os demais resultados para a mesma instância, poderá ser utilizado o melhor resultado como representante do conjunto de execuções, e assim compará-lo com os demais resultados obtidos pelo CPLEX.

Na execução do algoritmo, foram realizadas 30 execuções para cada conjunto de instâncias. Em seguida, o melhor resultado foi armazenado entre as execuções, juntamente com o tempo computacional correspondente. Além disso, foi calculada a média das soluções para cada instância e o *gap* entre a melhor solução encontrada e a melhor solução do CPLEX.

Inicialmente, para analisar o conjunto de soluções obtidas pelos algoritmos, optou-se por realizar a análise da distribuição da amostra dos resultados encontrados, utilizando um nível de significância igual a 5%. Foi realizado o teste estatístico desenvolvido por Shapiro & Wilk (1965), para testar a hipótese de normalidade dos dados, sendo este um teste de hipótese no qual existem sempre duas hipóteses: (i) a hipótese nula (H_0) e (ii) a hipótese Alternativa (H_a). Caso o resultado do p-valor seja maior que o nível de significância, a hipótese nula não é rejeitada; caso contrário, rejeita-se a hipótese nula. Para os casos em que a amostra não possua normalidade, foi utilizado o teste proposto por Wilcoxon (1992), sendo este teste realizado da mesma maneira que o teste *t-Student* porém, para amostras sem garantia de normalidade. A análise dos resultados dos testes *t-Student* e Wilcoxon (1992) utilizou a mesma metodologia adotada em Shapiro & Wilk (1965) quanto à verificação do p-valor e o nível de significância adotado.

Para a análise da distribuição amostral, a hipótese nula considera que os dados possuem uma distribuição normal. Caso a amostra apresente normalidade, será utilizado o teste *t-Student* (Gosset, 1908) para realizar comparações em pares e verificar se existe uma variação significativa entre os conjuntos de dados. A análise dos resul-

tados dos testes *t-Student* segue a mesma metodologia estabelecida por Shapiro & Wilk (1965) para a verificação do p-valor e a adoção do nível de significância.

Em resumo, a metodologia de análise seguirá o seguinte procedimento. Inicialmente, será avaliada a normalidade das 30 execuções de cada instância. Em seguida, será realizado um teste estatístico para determinar se o melhor resultado de cada instância difere significativamente da média do conjunto de resultados. Por fim, os melhores resultados das 30 execuções de cada instância serão comparados com os melhores resultados encontrados no CPLEX.

6.3 Resultados do Algoritmo ILS

O algoritmo *Iterated Local Search* (ILS) trabalha com os parâmetros $ILSmax$ e $VezesNivel$. Os valores desses parâmetros devem ser definidos previamente à execução dos experimentos computacionais completos. O parâmetro $ILSmax$ representa o número máximo de iterações; já o parâmetro $VezesNivel$ é responsável por atribuir a quantidade de iterações que são realizadas dentro do mesmo nível na execução corrente, antes de realizar a mudança de nível na solução corrente.

Estes parâmetros foram calibrados através do *software Iterated Racing for Automatic Algorithm Configuration* (IRACE) (López-Ibáñez et al., 2016). Foram realizados testes com conjuntos de instâncias com 10.000 e 50.000 clientes, raio de cobertura $R = 5.5$ e demanda a ser atendida $D = 50\%$. Os parâmetros $ILSmax$ e $VezesNivel$ foram testados, respectivamente, considerando valores nos seguintes conjuntos $\{50, 100, 150, 200, 250, 300\}$ e $\{2, 5, 10, 15, 20\}$. A configuração recomendada pelo $IRACE$ foi:

- $ILSmax = 100$
- $VezesNivel = 10$.

Foram realizados testes computacionais para instâncias com 10.000 e 50.000 clientes, do mesmo conjunto de instâncias utilizado no trabalho de Cordeau et al. (2019), variando o raio de cobertura e o percentual de demanda a ser atendida, mantendo sempre o número de 100 instalações em potencial. Os resultados obtidos e a análise estatística dos mesmos são apresentados na Tabela 6.1.

A primeira coluna mostra o número de clientes; a segunda coluna o percentual da demanda a ser atendido; a terceira coluna mostra o raio de cobertura. A quarta coluna apresenta o melhor valor de função objetivo encontrado pelo Algoritmo ILS ao solucionar a instância indicada, a quinta coluna apresenta a média de todas as 30 soluções encontradas para os testes realizados. A sexta coluna apresenta o resultado

do p-valor para o teste estatístico de Shapiro, no qual valores menores que $0,05$ rejeitam a hipótese nula, ou seja, não apresentam normalidade. Os resultados desta coluna que aparecem com hífen (-) representam instâncias em que todas as execuções apresentaram os mesmos resultados para todas as 30 execuções. Neste caso, não foi possível usar o teste, porém tais amostras são claramente não-normais. Como, para diversas instâncias, o teste não indicou normalidade no teste de Shapiro, foi aplicado o teste de Wilcoxon para verificar se o melhor resultado diverge da média das execuções. Os resultados do teste de Wilcoxon é apresentado na sétima coluna, em que é apresentado o p-valor do teste. Na oitava e última coluna é apresentado o resultado do teste da verificação da hipótese nula (H_0), que é sinalizado como "Ac" para as instâncias nas quais o melhor resultado não diverge das execuções, ou seja, aceita a hipótese nula, garantindo que o melhor resultado é um bom representante para ser usado como resultado daquela instância. Já os resultados desta coluna que são representados com um "R" rejeitam a hipótese nula, indicando que o melhor resultado não é um bom representante para os resultados daquela instância.

Através dos testes realizados verificou-se que a hipótese nula foi rejeitada quase na totalidade das instâncias analisadas e, portanto, os resultados encontrados possuem validade estatística e podem ser comparados com os resultados obtidos pelo CPLEX e também pelo outro método proposto neste trabalho.

Tabela 6.1: Resultados obtidos pelo algoritmo ILS e Testes Estatísticos para a solução de instâncias do Problema de Cobertura Parcial.

m	Dem.(%)	Raio	ILS		Análise Estatística		H_0
			FO_{melhor}	FO_{μ}	Shapiro	Wilcoxon	
10.000	50	5,50	74	74	-	-	Ac
		5,75	63	67	<0,05	0.876	Ac
		6,00	60	60	-	-	Ac
		6,25	48	58	<0,05	0.0017	Re
10.000	60	4,00	209	210	<0,05	0.647	Ac
		4,25	176	184	<0,05	0.869	Ac
		4,50	154	160	<0,05	0.382	Ac
		4,75	133	145	<0,05	0.0745	Ac
		5,00	120	120	-	-	Ac
10.000	70	3,25	529	548	0.12	-	Ac
		3,50	424	442	0.148	-	Ac
		3,75	324	337	<0,05	0.299	Ac
		4,25	241	253	0.556	-	Ac
50.000	50	5,50	82	90	<0,05	0.634	Ac
		5,75	76	76	-	-	Ac
		6,00	68	75	<0,05	0.033	Re
		6,25	66	67	<0,05	0.053	Ac
50.000	60	4,00	248	254	<0,05	0.933	Ac
		4,25	218	222	<0,05	0.772	Ac
		4,50	189	191	<0,05	0.620	Ac
		4,75	161	171	<0,05	0.482	Ac
		5,00	137	145	<0,05	0.9342	Ac
50.000	70	3,25	646	682	<0,05	0,821	Ac
		3,50	485	521	<0,05	0.578	Ac
		3,75	389	406	<0,05	0.635	Ac
		4,25	291	298	<0,05	-	Ac

6.4 Resultados do *Simulated Annealing*

Na implementação do algoritmo *Simulate Annealing*, os parâmetros $SAmax$ e α foram também calibrados usando o *software* *Iterated Racing for Automatic Algorithm Configuration* (IRACE). Para realizar a calibração pelo *IRACE*, também foram realizados testes com conjuntos de instâncias com 10.000 e 50.000 clientes, raio de cobertura $R = 5.5$ e demanda a ser atendida $D = 50\%$. Foram considerados valores para os parâmetros $SAmax$ e α, respectivamente, nos conjuntos $\{100, 150, 200, 250,300\}$ e $\{0,97, 0,98, 0,99\}$.

A configuração recomendada pelo *IRACE* é a que segue:

- $SAmax = 200$
- $\alpha = 0,99$.

Já os parâmetros utilizados para gerar a temperatura inicial foram $SAmax = 200$ e $\beta = 2$. O valor de $SAmax$ foi obtido pela calibração e o valor de β foi retirado da literatura.

Para a execução do algoritmo *Simulated Annealing*, foi mantido o mesmo conjunto de instâncias do trabalho de Cordeau et al. (2019) e os demais parâmetros referentes ao numero de clientes, instalações, raio de cobertura e demanda a ser atendida. Os resultados obtidos, bem como os testes estatísticos para cada instância, são apresentados na Tabela 6.2. Esta tabela está organizada conforme descrito na Seção Seção 6.3.

Através dos testes realizados, verificou-se que a hipótese nula foi aceita para todas as instância analisadas. Portanto, os resultados encontrados possuem validade estatística e podem ser comparados com os resultados de obtidos pelo CPLEX e também pelo outro algoritmo proposto neste trabalho.

Tabela 6.2: Resultados obtidos pelo algoritmo SA e Testes Estatísticos para a solução de instâncias do Problema de Cobertura Parcial.

m	Dem.(%)	Raio	SA		Análise Estatística		H_0
			FO_{melhor}	FO_{μ}	Shapiro	Wilcoxon	
10.000	50	5,50	74	80	<0,05	0.804	Ac
		5,75	63	75	<0,05	0.523	Ac
		6,00	61	64	<0,05	0.154	Ac
		6,25	48	61	<0,05	0.574	Ac
10.000	60	4,00	207	222	<0,05	0.565	Ac
		4,25	176	184	0.062	-	Ac
		4,50	154	157	<0,05	0.564	Ac
		4,75	133	139	<0,05	0.901	Ac
		5,00	120	120	-	-	Ac
10.000	70	3,25	517	549	<0,05	0.695	Ac
		3,50	408	431	0.259	-	Ac
		3,75	324	330	<0,05	0.773	Ac
		4,25	238	245	<0,05	0.773	Ac
50.000	50	5,50	82	86	<0,05	0.983	Ac
		5,75	76	77	<0,05	0.709	Ac
		6,00	68	71	<0,05	0.983	Ac
		6,25	66	66	-	-	Ac
50.000	60	4,00	250	262	<0,05	0.471	Ac
		4,25	217	225	<0,05	0.532	Ac
		4,50	185	198.5	0.25	-	Ac
		4,75	161	173	0.44	-	Ac
		5,00	137	146.5	<0,05	0.714	Ac
50.000	70	3,25	602	633	<0,05	0.918	Ac
		3,50	480	514	<0,05	0.95	Ac
		3,75	373	397	<0,05	0.82	Ac
		4,25	288	303	0.221	-	Ac

6.5 Comparação entre os Algoritmos Desenvolvidos

Nesta dissertação foram desenvolvidos dois algoritmos baseados em técnicas meta-heurísticas distintas. Para fins de comparação de qual técnica apresentou melhor desempenho, foi feita uma análise dos resultados.

A Tabela 6.3 apresenta a comparação entre os resultados obtidos pela aplicação do Algoritmo ILS e pelo solver *CPLEX*. Nesta tabela, a primeira coluna mostra o número de clientes; a segunda coluna o percentual da demanda a ser atendido; a terceira coluna mostra o raio de cobertura. A quarta coluna mostra o valor encontrado pelo solver CPLEX quando soluciona a instância indicada, enquanto a quinta coluna mostra o tempo de execução computacional associado, em segundos. A sexta coluna apresenta o valor encontrado pelo Algoritmo ILS ao solucionar a instância indicada, e sétima coluna mostra o tempo de execução computacional associado, em segundos. A oitava e última coluna mostra a diferença percentual entre os resultados obtidos pelo CPLEX e pelo algoritmo ILS proposto. O cálculo desta diferença (nomeada como *gap*) é dado por:

$$gap = 100 \left(\frac{fo(ILS) - fo(CPLEX)}{fo(ILS)} \right). \tag{6.1}$$

A Tabela 6.4 apresenta a comparação entre os resultados obtidos pela aplicação do algoritmo *Simulated Annealing (SA)* e o solver CPLEX. A disposição dos resultados nesta tabela é semelhante à disposição descrita para a Tabela 6.3. A oitava e última coluna mostra a diferença percentual entre os resultados obtidos pelo CPLEX e pelo algoritmo SA proposto. O cálculo desta diferença (nomeada como *gap%*) é dado por:

$$gap = 100 \left(\frac{fo(SA) - fo(CPLEX)}{fo(SA)} \right). \tag{6.2}$$

Na Tabela 6.5 é realizada uma análise entre os Algoritmos ILS e SA. A primeira coluna apresenta o número de clientes; a segunda coluna o percentual da demanda a ser atendido; a terceira coluna mostra o raio de cobertura; a quarta coluna apresenta o melhor valor de função objetivo encontrado nas 30 execuções do algoritmo SA; a quinta coluna mostra o tempo médio de todas as execuções; a sexta coluna mostra o melhor valor de função objetivo encontrado nas 30 execuções do algoritmo ILS; a sétima coluna mostra o tempo médio gasto para todas as execuções deste método; e, finalmente, a oitava coluna apresenta o *gap* percentual entre o melhor resultado do algoritmo SA e o melhor resultado do algoritmo ILS. O cálculo dos valores percentuais

Tabela 6.3: Comparação entre os resultados obtidos pela meta-heurística ILS (fo(ILS)) e os resultados obtidos pelo solver CPLEX (fo(CPLEX)) para a solução de instâncias do Problema de Cobertura Parcial.

N Clientes	Demanda(%)	Raio	fo(CPLEX)	Tempo CPLEX (seg)	fo(ILS)	Tempo (seg)	gap%
10.000	50	5,50	74.00	4.95	74.00	14.36	0.00
		5,75	63.00	5.66	63.00	13.08	0.00
		6,00	60.00	5.02	60.00	10.54	0.00
		6,25	48.00	2.34	48.00	8.38	0.00
10.000	60	4,00	207.00	15.13	209.00	58.52	0.97
		4,25	176.00	4.78	176.00	56.84	0.00
		4,50	154.00	8.72	154.00	43.74	0.00
		4,75	133.00	6.44	133.00	37.76	0.00
		5,00	120.00	7.31	120.00	30.97	0.00
10.000	70	3,25	517.00	3.42	529.00	212.42	2.32
		3,50	408.00	3.95	424.00	131.14	3.92
		3,75	324.00	7.84	324.00	152.66	0.00
		4,25	238.00	6.66	241.00	82.37	1.26
50.000	50	5,50	82.00	53.22	82.00	103.69	0.00
		5,75	76.00	47.36	76.00	126.66	0.00
		6,00	68.00	19.38	68.00	105.46	0.00
		6,25	66.00	213.14	66.00	71.45	0.00
50.000	60	4,00	248.00	183.23	248.00	577.92	0.00
		4,25	217.00	187.52	218.00	486.00	0.46
		4,50	185.00	133.92	189.00	433.48	2.16
		4,75	161.00	128.18	161.00	430.28	0.00
		5,00	137.00	130.08	137.00	297.31	0.00
50.000	70	3,25	611.00	321.25	646.00	1946.36	5.73
		3,50	478.00	210.75	485.00	1461.02	1.46
		3,75	373.00	138.14	386.00	1177.59	3.49
		4,25	284.00	256.63	291.00	877.35	2.46

de *gap* é dado por:

$$gap = 100 \left(\frac{fo(ILS) - fo(SA)}{fo(ILS)} \right). \tag{6.3}$$

Neste caso, o valor de $gap = 0$ representa que ambos algoritmos propostos encontraram o mesmo resultado. Valores de *gap* negativo indicam que o método ILS obteve soluções melhores que o SA; já valores de *gap* positivos indicam que o SA superou os resultados obtidos pelo ILS.

Tabela 6.4: Comparação entre os resultados obtidos pela meta-heurística SA (fo(SA)) e os resultados obtidos pelo solver CPLEX (fo(CPLEX)) para a solução de instâncias do Problema de Cobertura Parcial.

N Clientes	Demanda(%)	Raio	fo(CPLEX)	Tempo CPLEX (seg)	fo(SA)	Tempo (seg)	gap%
10.000	50	5,50	74.00	4.95	74.00	9.54	0.00
		5,75	63.00	5.66	63.00	9.23	0.00
		6,00	60.00	5.02	60.00	9.70	0.00
		6,25	48.00	2.34	48.00	9.46	0.00
10.000	60	4,00	207.00	15.13	207.00	22.36	0.00
		4,25	176.00	4.78	176.00	37.68	0.00
		4,50	154.00	8.72	154.00	40.31	0.00
		4,75	133.00	6.44	133.00	39.67	0.00
		5,00	120.00	7.31	120.00	39.45	0.00
10.000	70	3,25	517.00	3.42	517.00	39.99	0.00
		3,50	408.00	3.95	408.00	36.15	0.00
		3,75	324.00	7.84	324.00	32.15	0.00
		4,25	238.00	6.66	238.00	36.30	0.00
50.000	50	5,50	82.00	53.22	82.00	265.92	0.00
		5,75	76.00	47.36	76.00	327.91	0.00
		6,00	68.00	19.38	68.00	314.17	0.00
		6,25	66.00	213.14	66.00	316.83	0.00
50.000	60	4,00	248.00	183.23	250.00	305.15	0.81
		4,25	217.00	187.52	217.00	307.39	0.00
		4,50	185.00	133.92	185.00	316.58	0.00
		4,75	161.00	128.18	161.00	328.23	0.00
		5,00	137.00	130.08	137.00	328.17	0.00
50.000	70	3,25	611.00	321.25	615.00	300.19	0.65
		3,50	478.00	210.75	478.00	301.49	0.00
		3,75	373.00	138.14	373.00	304.67	0.00
		4,25	284.00	256.63	284.00	310.15	0.00

Tabela 6.5: Comparação entre os resultados obtidos pela meta-heurística SA (fo(SA)) e os resultados obtidos pela meta-heurística ILS (fo(ILS)) para a solução de instâncias do Problema de Cobertura Parcial.

N Clientes	Demanda(%)	Raio	fo(SA)	Tempo SA (seg)	fo(ILS)	Tempo ILS (seg)	gap%
10.000	50	5,50	74.00	9.54	74.00	14.36	0.00
		5,75	63.00	9.23	63.00	13.08	0.00
		6,00	60.00	9.70	60.00	10.50	0.00
		6,25	48.00	9.46	48.00	11.17	0.00
10.000	60	4,00	207.00	22.36	207.00	64.32	0.00
		4,25	176.00	37.68	176.00	61.14	0.00
		4,50	154.00	40.31	154.00	34.51	0.00
		4,75	133.00	39.67	133.00	45.10	0.00
		5,00	120.00	39.45	120.00	30.12	0.00
10.000	70	3,25	517.00	39.99	529.00	191.79	2.27
		3,50	408.00	36.15	423.00	152.26	3.55
		3,75	324.00	32.15	333.00	94.17	2.70
		4,25	238.00	36.30	246.00	70.04	3.25
50.000	50	5,50	82.00	265.92	82.00	149.28	0.00
		5,75	76.00	327.91	76.00	124.86	0.00
		6,00	68.00	314.17	75.00	106.95	9.33
		6,25	66.00	316.83	66.00	60.17	0.00
50.000	60	4,00	250.00	305.15	251.00	676.25	0.40
		4,25	217.00	307.39	217.00	358.55	0.00
		4,50	185.00	316.58	187.00	425.36	1.07
		4,75	161.00	328.23	162.00	578.32	0.62
		5,00	137.00	328.17	137.00	502.32	0.00
50.000	70	3,25	615.00	300.19	620.00	532.65	0.81
		3,50	478.00	301.49	478.00	687.51	0.00
		3,75	373.00	304.67	375.00	653.21	0.53
		4,25	284.00	310.15	284.00	761.25	0.00

6.6 Discussão dos Resultados

Foram solucionadas, para ambos os algoritmos, um total de 26 instâncias. No procedimento ILS, o valor de *gap* foi nulo em 20 destas instâncias, ou seja, em $68,97\%$ das instâncias o procedimento proposto encontrou a solução ótima. Dentre as 13 instâncias de 10.000 clientes avaliadas, em 9 (ou seja, $69,2\%$) destas o valor ótimo foi encontrado. Por outro lado, dentre as 13 instâncias envolvendo 50.000 clientes, o valor ótimo foi determinado para 7 instâncias ($53,8\%$). Ou seja, os resultados são de boa qualidade. Quanto ao tempo computacional, observou-se que os tempos gastos pelo CPLEX foram, em sua grande maioria, inferiores ao ILS. Uma possível explicação para o custo computacional mais elevado nestas instâncias é o uso da técnica de busca local do melhor vizinho, em que foi feita a exploração de todas as vizinhanças disponíveis, sendo, portanto, uma estrutura dispendiosa, mas que garante a determinação de melhor resultado na vizinhança avaliada.

Já o procedimento SA obteve bons resultados, com *gap* inferior a 1% para todas as instâncias analisadas. Dentre as 26 instâncias utilizadas, obteve-se um total de 24 soluções ótimas e somente 2 soluções com *gap* maior que zero. Vale ainda explicitar que, nestes casos, os *gaps* foram inferiores a 1%, demonstrando assim a eficiência do algoritmo para a solução do problema. Em contrapartida, somente para algumas soluções o SA obteve tempo inferior ao CPLEX. Isso se deve ao fato dos parâmetro $SAMax$ e α utilizarem valores mais elevados em sua calibração, o que proporcionou uma condição de resfriamento mais lenta, fazendo com que fossem aceitas mais soluções de qualidade inferior e ainda um maior número de iterações na execução do método.

Analisando as duas meta-heurísticas observou-se que o Algoritmo SA obteve soluções com qualidade superior às do Algoritmo ILS. Dentre as 26 instâncias analisadas, o SA obteve soluções superiores em 9 delas; nas demais soluções, os resultados se equivaleram. Já na comparação de tempo computacional a vantagem do SA sob o ILS permanece, com este método obtendo, dentre as 26 instâncias, 20 soluções com tempo mais rápido.

Capítulo 7

Conclusão

Neste capítulo são descritas, na Seção 7.1, as conclusões do trabalho. Na Seção 7.2 são discriminados os artigos oriundos da pesquisa realizada para a dissertação. Por fim, na Seção 7.3, são apresentadas propostas de trabalhos futuros.

7.1 Considerações Finais

Este trabalho teve, como seu foco, o problema de localização com cobertura parcial que tem, como objetivo, minimizar o custo de abertura de novas instalações, ao mesmo tempo em que deve garantir o atendimento de uma demanda mínima.

Foram propostos dois algoritmos meta-heurísticos. Em ambos os algoritmos as soluções iniciais foram geradas através do método de construção aleatória. Na implementação do *Iterated Local Search* foi aplicado o método VND para o refinamento das soluções. Já na implementação da meta-heurística *Simulated Annealing*, após a obtenção da solução inicial, o método foi aplicado para solução do problema.

Os algoritmos foram testados utilizando o mesmo conjunto de instâncias utilizados por Cordeau et al. (2019). Inicialmente, os parâmetros de cada algoritmo foram calibrados utilizando o *software Iterated Racing for Automatic Algorithm Configuration* (IRACE). Após a calibração foram realizados 30 testes computacionais para cada instância. Os resultados obtidos foram primeiramente validados estatisticamente e posteriormente comparados com os resultados obtidos pelo *solver* CPLEX. Por último, foi realizada uma comparação entre os resultados obtidos por cada método.

Os resultados obtidos indicam que o Algoritmo SA mostra-se mais eficaz quando comparado ao Algoritmo ILS. Quando comparado ao CPLEX, o Algoritmo SA mostrou-se ainda eficaz obtendo soluções ótimas na maioria dos testes e, por vezes, com tempos equivalentes ou próximo ao do CPLEX.

Os dois algoritmos concebidos foram fundamentados em técnicas de implementa-

ções simples, o que proporciona uma facilitação da replicação do método para aplicações em cenários reais, uma vez que o problema de cobertura parcial, apesar de pouco abordado na literatura, possui grande potencial de aplicação em casos reais. Podemos citar como exemplo o seu uso na implantação de redes de telecomunicação, em que podemos otimizar a colocação de antenas de celular 5G, de modo a garantir uma cobertura adequada em um percentual da área de interesse e otimizando os custos da instalação do sistema de telecomunicação. Devemos ainda citar a sua utilização na área de sensoriamento remoto, como monitoramento ambiental ou detecção de incêndios florestais, podendo ser utilizado para otimizar a colocação de sensores em uma região, pois a cobertura de toda a área de floresta é inviável financeiramente. Assim, através desta solução, temos a possibilidade de cobrir somente a área de interesse de forma eficiente.

7.2 Publicações

A realização deste trabalho resultou em 2 trabalhos:

- um artigo completo publicado no XIX Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2022)
- um artigo completo aceito para publicação no LV Simpósio Brasileiro de Pesquisa Operacional (SBPO).

Na primeira publicação foi apresentada o algoritmo *Iterated Local Search*, tratando um conjunto limitado das instâncias exploradas nesta dissertação. Já no segundo artigo foi apresentado o algoritmo baseado na meta-heurística *Simulated Annealing*, utilizando também um conjunto limitado das instâncias exploradas neste projeto, porém já utilizando a calibração dos parâmetros.

7.3 Trabalhos Futuros

Como trabalhos futuros, propõe-se:

(i) implementar um método exato para resolução deste problema;

(ii) implementar este problema sob condições de incerteza, trabalhando as incertezas na demanda a ser atendida ou no custo de abertura de novas instalações;

(iii) aplicar algoritmos genéticos evolutivos para o problema e verificar sua eficácia, comparado as meta-heurísticas de trajetórias utilizadas neste projeto.

Além disso, pode-se ainda realizar a implementação para problemas reais, tais como localização de antenas e/ou pontos de recargas para carros elétricos, sendo estes dois temas de grande relevância atual para a sociedade.

Referências

Bilal, N., Galinier, P., & Guibault, F. (2014). An iterated-tabu-search heuristic for a variant of the partial set covering problem. *Journal of Heuristics*, *20*, 143–164.

Cardoso, L. C., Mourao, F. P., de Sá, E. M., & de Souza, S. R. (2022). Metaheurística iterated local search aplicada ao problema de localização com cobertura parcial. In *Anais do XIX Encontro Nacional de Inteligência Artificial e Computacional* (pp. 142–151). SBC.

Church, R., & ReVelle, C. (1974). The maximal covering location problem. In *Papers of the regional science association* (pp. 101–118). Springer-Verlag.

Coco, A. A., Santos, A. C., & Noronha, T. F. (2018). Formulation and algorithms for the robust maximal covering location problem. *Electronic Notes in Discrete Mathematics*, *64*, 145–154.

Coco, A. A., Santos, A. C., & Noronha, T. F. (2022). Robust min-max regret covering problems. *Computational Optimization and Applications*, *83*, 111–141.

Cordeau, J.-F., Furini, F., & Ljubić, I. (2019). Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, *275*, 882–896. doi: 10.1016/j.ejor.2018.12.021.

Daskin, M. S., & Owen, S. H. (1999). Two new location covering problems: The partial p-center problem and the partial set covering problem. *Geographical Analysis*, *31*, 217–235.

Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M., & Goh, M. (2012). Covering problems in facility location: A review. *Computers & Industrial Engineering*, *62*, 368–407.

Galvão, R. D., & ReVelle, C. (1996). A lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, *88*, 114–123.

Goffe, W. L., Ferrier, G. D., & Rogers, J. (1994). Global optimization of statistical functions with simulated annealing. *Journal of econometrics*, *60*, 65–99.

Gosset, W. S. (1908). Student. *The Application of the'Law of Error'to the Work of the Brewery*, (pp. 3–6).

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations research*, *13*, 462–475.

Júnior, A. d. C. G., Souza, M. J. F., & Martins, A. X. (2005). Simulated annealing aplicado à resolução do problema de roteamento de veículos com janela de tempo. *TRANSPORTES*, *13*.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*, 671–680.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics* (pp. 320–353). Springer.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*, 1087–1092.

Mišković, S. (2017). A VNS-LP algorithm for the robust dynamic maximal covering location problem. *OR Spectrum*, *39*, 1011–1033.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, *24*, 1097–1100.

Montgomery, D. C. (2017). *Design and analysis of experiments*. John wiley & sons.

Osman, I. H., & Kelly, J. P. (1997). Meta-heuristics theory and applications. *Journal of the Operational Research Society*, *48*, 657–657.

Ribeiro, C. C. (1996). Metaheuristics and applications. *Advanced School on Artificial Intelligence, Estoril, Portugal*, .

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*, 591–611.

Sherali, H. D., Kim, S.-I., & Parrish, E. L. (1991). Probabilistic partial set covering problems. *Naval Research Logistics (NRL)*, *38*, 41–51.

Toregas, C., Swain, R., ReVelle, C., & Bergman, L. (1971). The location of emergency service facilities. *Operations research*, *19*, 1363–1373.

Vasko, F. J., Newhart, D. D., Stott Jr, K. L., & Wolf, F. E. (2003). A large-scale application of the partial coverage uncapacitated facility location problem. *Journal of the Operational Research Society*, *54*, 11–20.

Vatsa, A. K., & Jayaswal, S. (2021). Capacitated multi-period maximal covering location problem with server uncertainty. *European Journal of Operational Research*, *289*, 1107–1126.

Wilcoxon, F. (1992). *Individual comparisons by ranking methods*. Springer.