PRODUÇÃO ONLINE
REVISTA CIENTÍFICA ELETRÔNICA DE ENGENHARIA DE PRODUÇÃO
ISSN 1676-1901

ALGORITMOS PARA O PROBLEMA DE LOCALIZAÇÃO DE ESTAÇÕES DE CARREGAMENTO DE VEÍCULOS ELÉTRICOS

ALGORITHMS FOR THE LOCATION PROBLEM OF ELECTRIC VEHICLES CHARGING STATIONS

Yngrith Soares da Silva* E-mail: yngriths@gmail.com
Mário Mestria* E-mail: mmestria@ifes.edu.br
*Instituto Federal do Espírito Santo (IFES), Vitória, ES

Resumo: O Problema de Localização de Estações de Carregamento de Veículos Elétricos (PLECVE) é um problema importante no transporte, na logística e envolve a determinação de uma rede eficiente que garanta que todos os clientes acessem suas estações com a distância mínima de deslocamento. O problema é *NP*-difícil e complexo porque envolve várias restrições e a escala de aplicações do mundo real. Assim, os métodos heurísticos para resolver o PLECVE são fundamentais e esse trabalho apresenta algoritmos utilizando um modelo de otimização adaptado da literatura. O modelo seleciona um conjunto de estações de carregamento para alocar nos sites de localização candidatos. O objetivo é minimizar o custo fixo da estação de carregamento e o custo de viagem dos veículos elétricos. O modelo foi aplicado a uma região metropolitana usando viagens em linhas retas e as realistas. Os resultados mostraram que os algoritmos para o PLECVE encontraram soluções viáveis em um tempo computacional baixo. Os algoritmos propostos foram testados no banco de dados da literatura existente.

Palavras-chave: Veículos elétricos. Otimização. Metaheurística. Otimização por reações químicas. Algoritmo guloso.

Abstract: The Problem of Location of Electric Vehicle Charging Stations (PLEVCS) is an important problem in transportation, logistics and involves the determination of an efficient network that guarantees for all customers to access its stations with the minimum travel distance. The problem is *NP*-hard and complex because involving a number of the constraints and the scale of real-world applications. Thus, the heuristic methods to solve the PLEVCS are fundamental and this paper presents algorithms using an adapted optimization model from the literature. The model selects a set of charging stations to place from the candidate location sites. The objective is to minimize the charging station fixed cost and the electric vehicles travel cost. The model was applied to a metropolitan region using straight-line and realistic trips. The results showed that the algorithms for the PLEVCS found feasible solutions at a low computational time. The proposed algorithms were performed on the existing literature database.

Keywords: Electric vehicles. Optimization. Metaheuristics. Chemical Reaction Optimization. Greedy algorithm.

1 INTRODUÇÃO

Desde as décadas de 60 e 70, quando os veículos elétricos (VEs) ressurgiram, a taxa de inserção deles no mercado vem crescendo gradualmente (BNEF, 2017). A

preocupação com o meio ambiente e a dependência de combustíveis fósseis auxilia nesse aumento dos VEs.

Além disso, o motor de combustão interna está se tornando obsoleto, pois sua eficiência é de, aproximadamente, 30% da energia do combustível. Além da baixa eficiência, a emissão de gases poluentes, como o dióxido de carbono (CO_2), permanece alta (OSORIO, 2013).

É nesse cenário que os VEs, puros ou híbridos, vêm ganhando força como alternativa para a independência energética, pois os VEs têm a vantagem de produzir menos gases que são prejudiciais para a saúde do ser humano em comparação com os veículos de combustão interna, segundo Marinho (2013).

Outra vantagem oferecida pelos VEs é o custo por distância percorrida. Segundo a EDP Portugal (2017), o custo estimado de um carro elétrico é de 1315€/ano contra 2090 €/ano de um carro a combustão, supondo uma média de 12000 Km rodados/ano. Portanto, os VEs se tornarão os principais componentes importantes no sistema de transporte futuro.

Com o aumento da quantidade desses veículos surgem necessidades, como infraestrutura de recarga. Ainda existem poucas estações de carregamento disponíveis para atender a demanda de VEs e há a necessidade de um maior número de estações de carregamento em espaços públicos.

Com isso, surge o problema de localizar estas estações de carregamento em pontos estratégicos na cidade para atender a demanda dos clientes, para que não haja necessidade de os clientes deslocarem-se a uma distância muito longa para encontrar uma estação ou consigam encontrar uma estação de recarga sem que alcancem o limite mínimo de carga da bateria do VE (LAM; LEUNG; CHU, 2014).

Para localizar o ponto ótimo de instalação destas estações de carregamento podem ser usadas diferentes técnicas computacionais através de métodos exatos, como métodos *cutting planes, branch-and-bound e divide and conquer* (GENOVA; GULIASHKI, 2011). Os métodos exatos devem ser aplicados à resolução de problemas de instâncias pequenas, pois o tempo computacional cresce rapidamente à medida que aumenta o número de instâncias. Em situações reais, precisamos encontrar soluções com baixo tempo computacional.

Vários problemas em diversas áreas dos sistemas produtivos procuram a otimização de recursos, tempo mínimo de entrega de produtos, redução de custos

de produção e de estoques (FUCHIGAMI; MOURA; BRANCO, 2017) ou otimização de múltiplas respostas com conflitos, por exemplo, maximizar a qualidade de seu produto e ao mesmo tempo minimizar o impacto com o meio ambiente (GOMES *et al.*, 2017).

Riechi, Tormos e Hillebrand (2017) apresentaram um problema de otimização no qual buscaram encontrar o ponto de valor mínimo do custo médio anual de uma frota de ônibus e isso aprimorou o processo de tomada de decisão no planejamento da frota de uma empresa de transporte urbano.

Já em Barbosa *et al.* (2018) realizaram uma pesquisa e observaram diversos trabalhos da literatura com modelos de otimização sobre a evolução e as principais práticas utilizadas na gestão de custos logísticos nas organizações.

Em Bai *et al.* (2018), os autores desenvolveram algoritmos para o projeto de redes de serviços (que é o principal problema para o planejamento e otimização de redes de transporte de cargas, um problema *NP-Hard*).

No trabalho de Mestria (2016) foi proposto um algoritmo heurístico híbrido para resolver o Problema do Caixeiro Viajante com Grupamento (PCVG), outro problema *NP-Hard*, que usam várias estruturas de vizinhança variáveis, combinando a intensificação (usando operadores de busca local) e a diversificação (heurística construtiva e rotina de perturbação). O objetivo do PCVG é criar um ciclo Hamiltoniano, isto é, visitar todos os vértices com custo mínimo, onde os vértices são particionados em clusters e todos os vértices de cada cluster devem ser visitados de forma contígua.

No trabalho de Lam e Li (2010), o problema de localização de estações de carregamento de veículos elétricos foi solucionado a partir de quatro modelos propostos e a comparação entre eles. Os métodos apresentaram características próprias e foram adequados para diferentes situações. Lam e Li (2010) estudaram os locais onde as estações de carregamento deviam ser construídas em uma cidade para poder minimizar o custo de construção com cobertura da cidade inteira e o cumprimento da conveniência para o motorista.

No trabalho de Baouche *et al.* (2014), o método de localização de estações de carregamento é baseado em uma adaptação de dois modelos de localização clássicos: *Fixed Charge Location* e *p*-Dispersão. Ao invés da fixação das estações de carregamento nos pontos de demanda, o modelo dos autores se concentra em

minimizar o custo total da viagem para chegar ao local da estação e o custo de investimento da própria estação de recarga. A demanda de energia foi derivada de um modelo de consumo de veículo desenvolvido como parte da biblioteca VEHLIB. O consumo de energia foi avaliado pelo VEHLIB a partir de viagens reais.

Em outubro de 2018, o BNDES (Banco Nacional de Desenvolvimento Econômico e Social) liberou um aporte financeiro de 6,7 milhões de reais para dois projetos de estações de recarga de VEs (UOL, 2018).

Como observado acima, diversos trabalhos da literatura procuram realizar a otimização de alguma métrica, inclusive a localização de estações de carregamento, estabelecendo modelos matemáticos e sendo resolvidos através de algoritmos com diversas técnicas.

Este artigo tem como objetivo a apresentação do algoritmo guloso e de um algoritmo baseado na metaheurística CRO para determinar a localização do menor número possível de estações de carregamento de veículos elétricos. Assim, procuraremos atingir soluções ótimas ou de alta qualidade nesse trabalho com intuito de otimizar o valor da função objetivo de um modelo matemático proposto e adaptado de Baouche *et al.* (2014).

O artigo é estruturado da seguinte forma: na segunda seção tem-se uma revisão bibliográfica abordando o problema de localização de facilidades, algoritmo guloso e metaheurística CRO. Na terceira seção é mostrada a metodologia, a formulação matemática do problema e o levantamento de dados realizado. Na quarta seção, os resultados e as discussões são apresentados e na última seção as considerações finais são realizadas.

2 REVISÃO BIBLIOGRÁFICA

2.1 Problema de localização de facilidades

O problema de localização de facilidades é um método dentro da temática da Pesquisa Operacional de grande importância, pois uma boa localização de facilidades pode garantir melhorias na qualidade de serviços e a competitividade de um negócio, diminuindo os custos de logística (OLIVEIRA, 2012).

Trata-se da decisão de localizar objetos, chamados de facilidades, para que atendam da melhor forma aos critérios de demanda. Este problema também é conhecido como problema de localização-alocação (OLIVEIRA, 2012).

Podem ser consideradas facilidades: redes de comunicação, subestações em rede elétrica, estações de carregamento de veículos elétricos.

Esta teoria avançou a partir das pesquisas de Hakimi (1964) com as formulações do problema de *p*-mediana, que tem como objetivo localizar *p* facilidades para atender *n* pontos de demanda de forma que a distância percorrida do ponto de demanda a facilidade seja a menor possível (TRAGANTALERNGSAK *et al.*,1999). O problema de *p* medianas pode ser classificado em: capacitado e não-capacitado. No capacitado, cada local candidato a se tornar facilidade tem uma capacidade finita, não podendo atender clientes além do que a sua capacidade permite. Já no problema não-capacitado, o local candidato a facilidade pode atender um número infinito de clientes (TRAGANTALERNGSAK *et al.*, 1999).

Os problemas de localização de facilidades são considerados NP-*hard*, ou seja, de difícil solução, devido a sua complexidade (DASKIN, 2015). Esses problemas são de natureza combinatória. Existem diversas formas de solucioná-lo, como os métodos exatos, que podem ser aplicados a problemas de pequeno porte, já que buscam o ponto ótimo (STEINER, 2003).

Para a resolução de problemas de grande porte, na qual a utilização de um método exato não é adequada devido ao tempo de processamento, aplicam-se métodos aproximados que fornecem soluções próximas da ótima em um tempo viável, como a heurística relaxação Lagrangeana/Surrogate (COSTA, 2005). Contudo, a maioria das heurísticas é utilizada para a resolução de um problema específico, não sendo eficiente na resolução de problemas genéricos (SOUZA, 2008).

Então, outra forma de solucionar esse tipo de problema é utilizar metaheurísticas, que são métodos de busca inteligentes que se adaptam a diversos tipos de problemas e permitem obter uma solução aproximada do problema em um bom tempo computacional (HÖRNER, 2009).

Temos como exemplos de técnicas metaheurísticas: GRASP (procedimento de pesquisa gulosa, aleatória e adaptativa) (FEO e Resende, 1995), *Simulated Annealing* (CHIYOSHI e GALVÃO, 2000), Busca em Vizinhança Variável (*Variable*

Neighborhood Search) (HANSEN *et al.*, 2001), Algoritmos Genéticos (CORREA *et al.*, 2001), CRO (*Chemical Reaction Optimization*) (Lam e Li, 2010), além do algoritmo guloso (*Greddy Algorithm*) (ROCHA, 2004), que pode ser utilizado dentro de módulos de metaheurísticas.

2.2 Algoritmo Guloso

Uma boa forma de resolução de problemas de otimização está em usar os algoritmos gulosos, que sempre escolhem a melhor solução naquele instante para chegar a um ótimo global. Progridem de cima para baixo, isto é, a cada iteração, a instância do problema é reduzida (ROCHA, 2004).

É um algoritmo determinístico, ou seja, dado um conjunto de entrada, sempre produzirá a mesma solução com este conjunto. Uma grande vantagem do algoritmo guloso é o reduzido tempo computacional. Em compensação, a solução obtida nem sempre será a melhor que pode ser obtida (BASTOS, 2004).

A cada decisão a ser tomada, sempre escolhe a alternativa que parece ser a mais promissora naquele dado momento, ou seja, escolhe o ótimo local. Além disso, uma escolha feita nunca é revista (SILVA et al., 2015).

A Figura 1 ilustra o funcionamento de um algoritmo guloso genérico (ROCHA, 2004).

Figura 1 - Pseudocódigo do algoritmo guloso

```
1:  função ALGORITMOGULOSO(C: conjunto)          ▷ C é o conjunto de candidatos
2:      S ← ∅                                    ▷ S é o conjunto que irá conter a solução
3:      enquanto C ≠ ∅ e não solução(S) faça
4:          x ← seleciona C
5:          C ← C {x}
6:          se é viável S ∪ {x} então
7:              S ← S ∪ {x}
8:          fim se
9:      fim enquanto
10:     se solução(S) então
11:         retorne S
12:     senão
13:         retorne "Não existe solução!"
14:     fim se
15: fim função
```

Fonte: Rocha (2004)

Neste trabalho, a escolha aleatória de um grupo de candidatos para o algoritmo guloso foi feita a partir de uma escolha randômica destes locais. Para isto, foi utilizado a função rand( ).

Depois de escolhidos, os candidatos passaram por uma seleção na qual foi feita a soma total das distâncias do local candidato selecionado até os outros locais. Depois de obtida a soma total das distâncias de todos os candidatos escolhidos, foi utilizado um algoritmo de ordenação, o *Quicksort* (SEDGEWICK, 1983). A ordenação foi feita em ordem crescente.

O *Quicksort* foi escolhido como método de ordenação por ser um dos mais rápidos dentre os algoritmos conhecidos, o que mantém o tempo computacional do Algoritmo Guloso baixo. O candidato com o menor valor de soma vai para o conjunto solução. Os outros são descartados e vão para o conjunto de candidatos rejeitados. Isto é feito até a obtenção do valor de candidatos no conjunto solução que correspondam ao número de medianas (*p*).

2.3 Metaheurística CRO

É uma metaheurística probabilística inspirada na natureza das reações químicas. Ou seja, se baseia na estrutura da molécula, que é composta de vários átomos (BARHAM et al., 2016).

A principal característica das reações químicas é transformar substâncias instáveis em estáveis. Numa visão microscópica, a reação química transforma o estado de energia excessivo das moléculas em um estado de energia potencial mínima, que sustente a existência da molécula.

O sistema de reação química é representado pelas substâncias químicas e suas redondezas. As substâncias químicas possuem energia potencial e cinética. Já a redondeza tem sua energia representada por um reservatório de energia central, denominado *buffer*, no CRO (LAM E LI, 2012).

As reações podem ser endotérmicas, quando necessitam de calor do meio para iniciar a reação, ou exotérmicas, quando liberam calor para o meio. Estes dois tipos de reações são caracterizados no CRO pelo tamanho do buffer. Se for positivo, a reação é endotérmica. Se for negativo, a reação é exotérmica (LAM E LI, 2012).

Existem quatro tipos de reações químicas: colisão ineficaz na parede, decomposição, colisão ineficaz intermolecular e síntese (BARHAM et al., 2016).

O algoritmo baseado na metaheurística CRO foi implementada em linguagem C++ orientada a objeto, visto que as moléculas podem ser instâncias de uma classe com seus atributos e as reações elementares são métodos de uma classe Molécula.

A estrutura molecular (ω) possui uma solução do problema. Nesse caso, o valor da função objetivo e as estações escolhidas. Na colisão ineficaz na parede, ω é perturbada, transformando-se em ω', que pertence à vizinhança de ω.

Na colisão ineficaz intermolecular ($\omega_1 + \omega_2 \rightarrow \omega'_1 + \omega'_2$), ω'_1 e ω'_2 também são obtidos através do algoritmo guloso, como na colisão ineficaz na parede.

Na decomposição ($\omega \rightarrow \omega'_1 + \omega'_2$), ω'_1 e ω'_2 são gerados aleatoriamente no espaço solução através da modificação do algoritmo guloso, onde o laço de repetição é iterado um número aleatório de vezes e é selecionado um nó aleatório, como proposto por Lam, Leung e Chu (2014). O mesmo é feito para a síntese ($\omega_1 + \omega_2 \rightarrow \omega'$).

A frequência de ocorrência da decomposição e síntese pode ser influenciada indiretamente alterando-se os parâmetros do CRO denominados α e β, respectivamente.

Foram criadas as variáveis correspondentes aos parâmetros: PopSize, KELossRate, MoleColl, buffer, InitialKE, α e β. Depois foi definida a classe "Molécula"

com seus atributos ω, PE, KE, NumHit, MinStruct, MinPE e MinHit, e os quatro métodos correspondentes às reações elementares.

Como ponto de partida para o algoritmo CRO, utilizamos os valores dos parâmetros mostrados em Lam, Leung e Chu (2014). Contudo, cada aplicação tem uma configuração de parâmetros própria, devido aos objetivos mais específicos de cada aplicação e como nossas instâncias são diferentes das instâncias do artigo citado, tivemos que variar os parâmetros do algoritmo CRO desenvolvido.

Os parâmetros utilizados para o caso real foram: KELossRate= 0.5; MoleColl= 0.5; InitialKE= 1000; buffer= 0; alfa= 500; beta= 1000 e PopSize= 10. Já para resolução dos dados da base da literatura foram: KELossRate= 0.9; MoleColl= 0.3; InitialKE= 10000; buffer= 0; alfa= 50; beta= 1000 e PopSize= 10.

Os valores citados acima foram modificados de Lam, Leung e Chu (2014) a partir da alteração de um único parâmetro de cada vez e visualização dos resultados obtidos com valores da base OR-Library. O primeiro valor alterado foi KELossRate e por último, o valor de beta.

Os critérios de parada definidos foram o número de iterações sem melhorias e número de iterações totais. Os valores foram estipulados em 300 e 5000, respectivamente.

3 METODOLOGIA

Nessa seção apresentaremos os algoritmos para solucionar o problema de localização de estações de carregamento de veículos elétricos. Todos os algoritmos foram programados em linguagens estruturadas. O algoritmo guloso foi desenvolvido em linguagem C. A metaheurística CRO foi desenvolvida em C++. Para os testes foi utilizado uma máquina de 64 bits, processador Intel Core i5 de 2,4GHz, 8GB de memória e Windows 10.

Neste trabalho, foram utilizados o algoritmo guloso e a metaheurística CRO para resolver um caso real da região metropolitana de Vitória-ES, assim como foram testados para resolução de 40 instâncias conhecidas da biblioteca *OR-Library* (BEASLEY, 2017).

Foi escolhido o algoritmo guloso para a resolução destes problemas por ter um baixo tempo computacional comparado a outros algoritmos e produzir soluções com diferença mínima de qualidade comparada ao valor ótimo.

Já a metaheurística CRO foi usada porque tem como vantagens: resolver diferentes problemas de natureza combinatória, ter potencial para solucionar problemas que não tenham sido resolvidos por outras metaheurísticas, além de possuir as vantagens das metaheurísticas *Simulated Anneling* e do Algoritmo Genético.

3.1 Formulação Matemática

O problema da *p*-mediana utilizado é uma adaptação do problema proposto por Baouche *et al.*(2014), visando minimizar o custo da localização das estações de carregamento e a distância percorrida pelos clientes até uma estação. É importante formalizar matematicamente esse problema, pois os algoritmos desenvolvidos derivam desta formulação. O modelo utilizado possui a seguinte função objetivo, restrições e parâmetros:

Variáveis de decisão:

$$x_j = \{1 \; se \; a \; estação \; candidata \; j \; foi \; selecionada; \; ou \; 0, do \; contrário \quad (1)$$

$$y_{ij} = \{1 \; se \; o \; centro \; de \; demanda \; i \; é \; coberto \; pela \; estação \; j; \; ou \; 0, do \; contrário \quad (2)$$

Modelo:

$$Min \sum_{j \in J} fj.xj + \alpha \sum_{i \in I} \sum_{j \in J} d_{ij} . y_{ij} \quad (3)$$

Com as seguintes restrições:

$$\sum_{j \in J} y_{ij} = 1, \; \forall i \in I \quad (4)$$

$$\sum_{j \in J} y_{ij} = p, \; \forall i \in I \quad (5)$$

$$y_{ij} - x_j \leq 0 \; ; \; \forall i \in I; \; \forall j \in J \quad (6)$$

$$x_j \in \{0,1\}; \; j \in J, \; r \geq 0 \quad (7)$$

$$y_{ij} \in \{0,1\};\ i \in I, j \in J \quad (8)$$

onde:

I: Conjunto de centros de demanda
J: Conjunto de locais candidatos a estações de recarga
f_j: Custo para localização da estação de recarga
d_{ij}: Distância, percorrida pelo veículo, de um centro de demanda *i* até a estação de recarga *j*
α: Custo do Kilowatt hora
p: Número de estações escolhidas (medianas)

A função objetivo (3) é a função que minimiza o custo total de instalação das estações de carregamento e a distância de cada cliente ao local da estação candidata escolhida. A restrição (4) diz que toda demanda do centro de demanda deve ser satisfeita. A restrição (5) estabelece que o número de estações escolhidas seja igual a *p*. A restrição (6) assegura que um centro de demanda só pode ser associado a uma estação escolhida. As restrições (7) e (8) mostram que as variáveis de decisão são inteiras binárias.

Como o objetivo deste trabalho é avaliar os algoritmos para garantir que todos os usuários possam acessar as estações de carregamento, foi utilizado o caso da *p*-mediana puro.

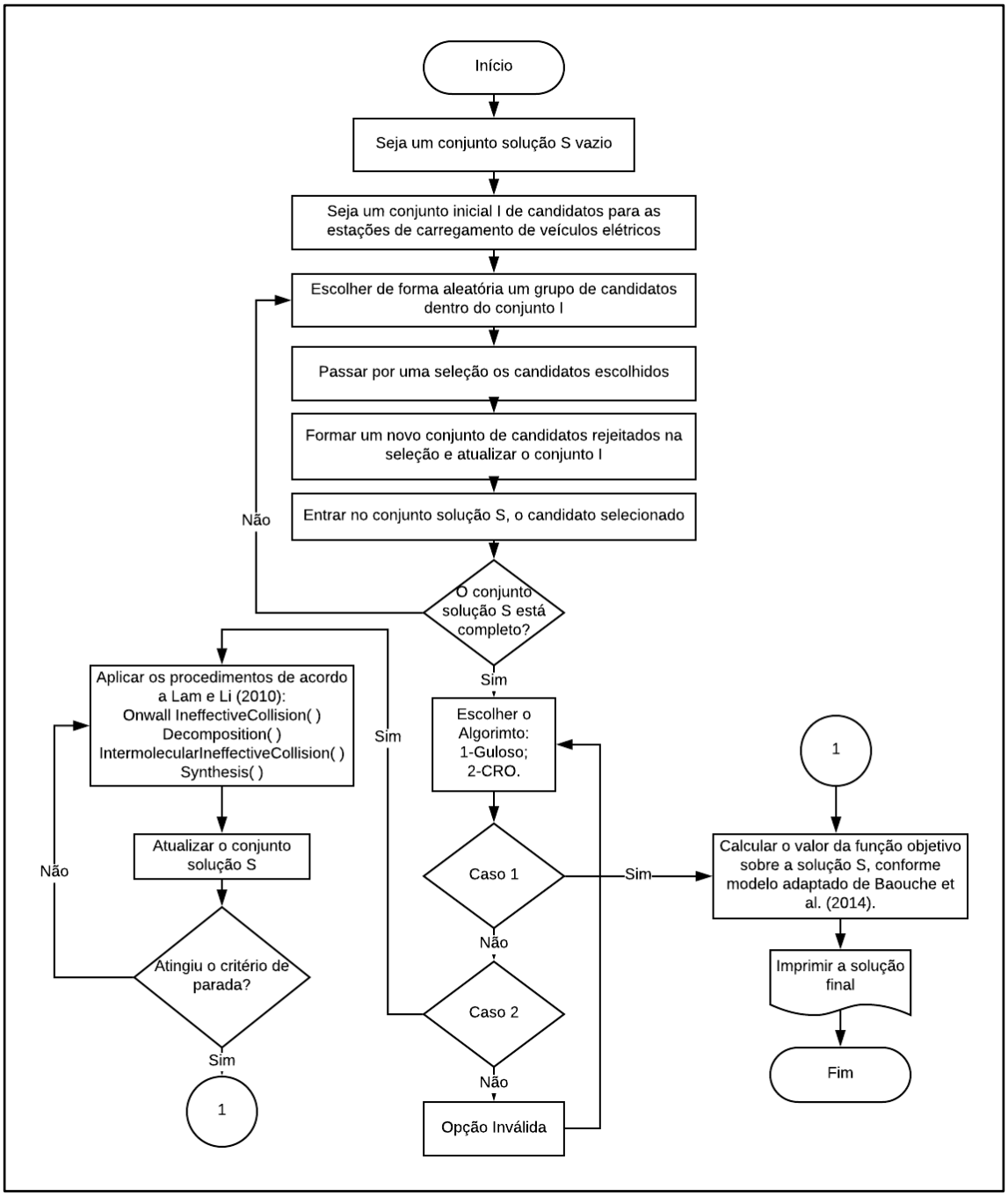
Então, o custo (*fj*) de localização de uma estação de carregamento *j* foi mantido fixo, pois foi considerado que o custo de instalação é igual para todas as estações, independentemente da localização (custo monetário do espaço territorial), e ainda foi considerado que as estações possuem o mesmo número de baias (pontos de recarga do veículo elétrico). Assim, a função (3) torna-se:

$$Min \quad \alpha \sum_{i \in I} \sum_{j \in J} d_{ij} . y_{ij} \quad (9)$$

Também foi considerado que o custo do quilowatt hora é α = 1, devido ser uma constante e não impactar na localização das estações de trabalho.

Na Figura 2, apresentamos um fluxograma para determinar a localização do menor número possível de estações de carregamento de veículos elétricos, mostrando o objetivo desse trabalho de forma concisa.

Figura 2 – Fluxograma para determinar a localização das estações de carregamento de veículos elétricos



Fonte: Elaborado pelo autor (2018)

3.2 Levantamento de Dados

3.2.1 Casos reais - Vitória

Para a resolução do caso real, foi escolhida a região metropolitana de Vitória, ES, pois é a que possui o maior número de automóveis em circulação do Estado, segundo DENATRAN (2015).

Primeiramente, definimos a localização dos locais candidatos a estações e dos clientes. Foram selecionados 8 locais candidatos e 16 clientes na região de Vitória-ES.

Os locais candidatos escolhidos são locais de grande circulação de automóveis na cidade, como centros comerciais, instituições de ensino e áreas de lazer. Além de serem espaços nos quais, geralmente, ocorre um tempo de permanência mínima superior a 15 minutos. Estes locais estão listados na Tabela 1.

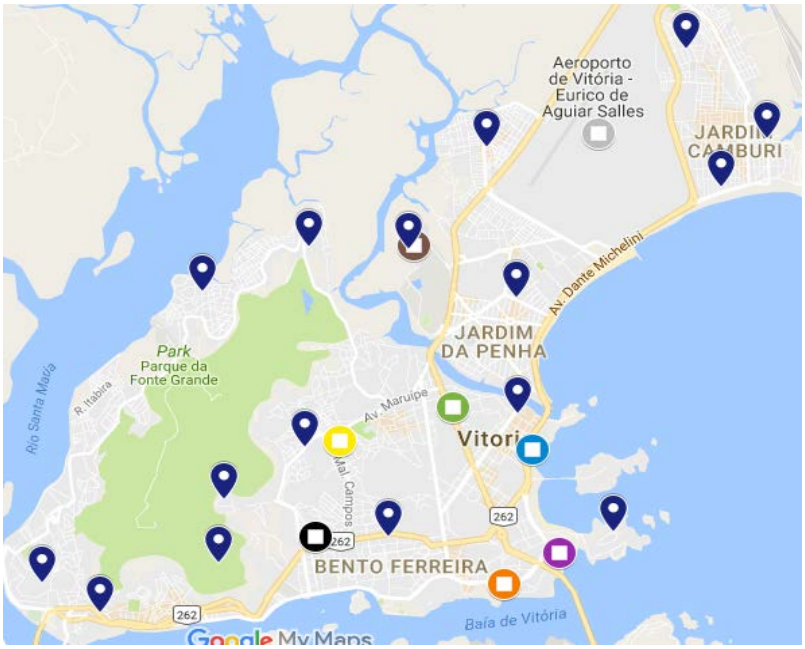
Tabela 1 - Locais candidatos a estações de carregamento em Vitória

Número	Local	Cor do ícone*
1	OK Hipermercado	Verde
2	Shopping Vitória	Roxo
3	Aeroporto de Vitória	Cinza
4	UFES Goiabeiras	Marrom
5	UFES Maruípe	Amarelo
6	IFES Vitória	Preto
7	Praça dos Namorados	Azul
8	Praça do Papa	Laranja

Fonte: Elaborado pelo autor (2018)
* Ver detalhes na Figura 1

Dado que os clientes podem estar em qualquer região da cidade de Vitória, estes foram selecionados de forma aleatória, mas cobrindo a maior área possível da região, como mostra a Figura 3.

Figura 3 – Locais candidatos a estações e os clientes escolhidos em Vitória



Fonte: Elaborado pelo autor (2018)

Os mapas mostrando as configurações escolhidas foram plotados no Google My Maps (Google Maps, 2017), para melhor visualização. Na Figura 1, os ícones em azul são os clientes e os ícones quadrados, são os locais candidatos, como descrito na Tabela 1.

Para a região de Vitória foram realizados dois tipos de testes. O primeiro com distâncias em linha reta e o segundo, com distâncias reais. Para o teste com distâncias em linha reta, utilizou-se as coordenadas geográficas encontradas no Google My Maps, mostradas nas tabelas 2 e 3.

Tabela 2 – Longitude e latitude dos locais candidatos mostrados no Google Maps

Locais candidatos	Latitude	Longitude
Ok Hipermercado	-20,29359	-40,30146
Shopping Vitória	-20,31242	-40,28781
Aeroporto	-20,25802	-40,28297
Ufes Goiabeiras	-20,27252	-40,30641
Ufes Maruípe	-20,29784	-40,31586
IFES Vitória	-20,31019	-40,31891
Praça dos Namorados	-20,29892	-40,29140
Praça do Papa	-20,31638	-40,29495

Fonte: Elaborado pelo autor (2018)

Tabela 3 – Longitude e latitude dos clientes mostrados no Google Maps

Locais candidatos	Latitude	Longitude
Cliente 1	-20,31377	-40,33135
Cliente 2	-20,24724	-40,27173
Cliente 3	-20,27869	-40,33330
Cliente 4	-20,27310	-40,30708
Cliente 5	-20,31031	-40,30961
Cliente 6	-20,27945	-40,29338
Cliente 7	-20,29881	-40,32027
Cliente 8	-20,30548	-40,33060
Cliente 9	-20,32008	-40,34638
Cliente 10	-20,31625	-40,35377
Cliente 11	-20,30976	-40,28100
Cliente 12	-20,26503	-40,26727
Cliente 13	-20,25988	-40,29720
Cliente 14	-20,29447	-40,29317
Cliente 15	-20,27286	-40,31978
Cliente 16	-20,25873	-40,26143

Fonte: Elaborado pelo autor (2018)

Os valores da latitude e longitude fornecidos pelo Google My Maps estão em graus decimais. Portanto, foi necessário a conversão dos valores de latitude e longitude para cálculo destas distâncias. Então, converteu-se todas as latitudes e longitudes de graus decimais para graus, minutos e segundos.

Então, calcula-se a distância latitudinal (DLA) e a distância longitudinal (DLO). Os resultados das distâncias latitudinal e longitudinal são multiplicados por uma milha náutica. Por convenção, milha náutica equivale a 1852 metros.

Após, é feito o cálculo de distância euclidiana. A distância euclidiana (DE) entre dois pontos A e B quaisquer é dada por:

$$DE_{AB} = [(DLA_b - DLA_a)^2 + (DLO_b - DLO_a)^2]^{1/2} \quad (10)$$

Um exemplo deste cálculo está mostrado no apêndice B.

A partir destes cálculos, foi implementada uma rotina no Dev C++ 2018 para conversão destes valores e o preenchimento da matriz de distâncias. O Google My Maps fornece a distância em linha reta entre os locais, mas se torna inviável coletar estas informações para grandes instâncias.

No teste com distâncias reais, a matriz de distâncias foi preenchida com a menor distância percorrida de automóvel pelas ruas da cidade de todos os clientes até os locais candidatos. Essa distância é mostrada pelo Google Maps.

Para o algoritmo guloso, só é necessário realizar um teste (uma execução) para cada valor de mediana (quantidade de estações escolhidas), pois é um algoritmo determinístico. Como *p* varia de 1 a 4, no total foram realizados 4 testes com o algoritmo Guloso para distâncias euclidianas.

Para a metaheurística CRO, foi realizada uma rotina de 15 testes (15 execuções) para cada valor de mediana, para distâncias em linha reta. Isso se faz necessário, pois o CRO é probabilístico. Neste caso, *p* variou de 1 a 4, então foram realizados 60 testes no total. A quantidade de 15 testes foi definida empiricamente e com base em valores utilizados nas literaturas consultadas.

Para distâncias reais com o algoritmo guloso, foi executada também uma rotina de 4 testes. Para o teste com distâncias reais com o CRO, foi realizada a mesma rotina explicada acima.

Nos dois casos foram encontrados os locais candidatos escolhidos, o número de iterações realizadas, o valor da função objetivo e o tempo de execução. Os detalhes serão mostrados na seção resultados.

3.2.2 Base de dados *OR-Library (Operational Research Library)*

A base de dados da biblioteca *OR-Library* de Beasley (2017) é altamente utilizada na literatura e para provar a eficácia e verificar a qualidade dos algoritmos desenvolvidos, foram realizados testes com as 40 instâncias desta biblioteca.

Existem 40 arquivos de dados em formato `txt` para problemas de *p*-mediana não capacitada. Os arquivos contêm entre 100 e 900 vértices (consumidores) e de 5 a 200 medianas (locais candidatos às estações). Os dados fornecem, neste caso, a distância de um vértice a outro.

Os dados nestes arquivos não estão completos, então Beasley (2017) sugere que seja utilizado o algoritmo de Floyd para obtenção da matriz completa, porque é um tipo de algoritmo que calcula o menor caminho entre todos os pares de vértices de um grafo e possui complexidade (n^3).

A partir do pseudocódigo, mostrado na Figura 4, foi desenvolvido uma rotina no Dev C++ 2018 para preenchimento de todas as distâncias nas matrizes.

Figura 4 – Pseudocódigo do algoritmo de Floyd

```
Seja n o número de vértices
Faça c(i,j) = infinito para i= 1,....,n e j= 1, .....,n.
c(i,j) = 0 para i= 1, ...., n
Faça ler cada linha do arquivo txt
Sendo os três números da linha de dados i, j, k então
Faça c(i,j)= k e c(j,i)=k
```

Fonte: Elaborado pelo autor (2018)

O algoritmo irá preencher a matriz com zero nos casos em que *i=j*. Quando *i* e *j* forem diferentes, mas os valores forem conhecidos da base, os mesmos serão preenchidos com esse valor. Caso contrário, serão preenchidos com um valor considerado infinito. Neste trabalho, foi utilizado como infinito (INT_MAX)/2 no software desenvolvido.

4 RESULTADOS

4.1 Casos Reais – Vitória

Os resultados obtidos com o algoritmo guloso para a cidade de Vitória estão dispostos nas Tabelas 4 e 5. As tabelas mostram o número de estações de carregamento (medianas), as estações escolhidas, o número de iterações, o valor da função objetivo e tempo de execução do algoritmo.

Tabela 4 - Resultados do algoritmo Guloso para 16 clientes e linha reta em Vitória

Número de estações (p)	Estações escolhidas	Iterações	Função objetivo(km)	Tempo (s)
1	1	440	221,9	3
2	1 e 4	468	178,4	3
3	1, 4 e 5	480	115,9	6
4	1,4, 5 e 7	524	57,03	3

Fonte: Elaborado pelo autor (2018)

Tabela 5 - Resultados do algoritmo Guloso para 16 clientes e distâncias reais em Vitória

Número de estações (p)	Estações escolhidas	Iterações	Função objetivo(km)	Tempo (s)
1	5	312	366,085	4
2	5 e 7	340	271,764	4
3	1, 5 e 7	352	179,445	4
4	1,5, 6 e 7	396	87,681	9

Fonte: Elaborado pelo autor (2018)

Nos testes realizados para Vitória com o algoritmo Guloso, mostrados nas Tabelas 2 e 3, verificamos que a estação escolhida quando p=1, utilizando a matriz de distâncias em linha reta e de distâncias reais, é diferente.

Para p=2, vemos que o resultado obtido nos dois casos também é diferente. Então, para p=1 e p=2 houve discrepância nos resultados. Nos resultados do teste para p=3, as estações 1 e 5 são escolhidas nas duas configurações. Nos resultados para escolha de 4 estações (p=4), as estações 1, 5 e 7 são obtidas nas duas configurações. Analisando o número de iterações na configuração de distâncias em linha reta e distâncias reais, vemos que o número de iterações aumenta em ambos os casos com a quantidade de estações escolhidas.

Não foi possível encontrar um padrão de comportamento para o tempo de execução do algoritmo com o aumento do número de medianas, mas variou no intervalo de 3 a 9s.

Observando os valores calculados para as funções objetivo, vemos que diminuem com a quantidade de estações escolhidas, sendo maior para distâncias reais, como era esperado.

Nos testes realizados para Vitória com o algoritmo baseado na metaheurística CRO, mostrado na Tabela 6, verificamos que o resultado difere do encontrado para o algoritmo Guloso para p=3 e p=4, pois o algoritmo CRO encontrou uma solução com menor valor da função objetivo. Para as configurações de 1 e 2 estações, o resultado encontrado foi igual ao obtido para o algoritmo guloso. Com este algoritmo, foi possível observar que o tempo computacional aumentou com a quantidade de estações. Foram mostrados os melhores valores obtidos para a função objetivo das

15 execuções realizadas para cada mediana (número de estações) e o tempo mostrado é a média dos tempos obtidos nos 15 testes.

Tabela 6 - Resultados da metaheurística CRO para 16 clientes e linha reta em Vitória

Número de estações	Estações escolhidas	Função objetivo (melhor)(Km)	Tempo médio (s)
1	5	221,9	4
2	1 e 5	178,4	6
3	1, 4 e 5	107,0	9
4	1, 4, 5 e 7	27,40	11

Fonte: Elaborado pelo autor (2018)

Nos testes realizados para distâncias reais com o algoritmo CRO, mostrado na Tabela 7, verificamos que o resultado difere do encontrado para o algoritmo Guloso para p=2, p=3 e p=4, pois o CRO encontrou uma solução com menor valor da função objetivo. Para a configuração de 1 estação escolhida, o resultado encontrado foi igual ao obtido para o algoritmo guloso. Também foram mostrados os melhores valores obtidos para a função objetivo dos 15 testes realizados e o tempo mostrado é a média dos tempos obtidos nos 15 testes.

Tabela 7 - Resultados da metaheurística CRO para 16 clientes e distâncias reais em Vitória

Número de estações	Estações escolhidas	Função objetivo (melhor)(Km)	Tempo médio (s)
1	5	366,08	3
2	5 e 6	243,4	6
3	1, 5 e 6	168,0	6
4	1, 4,5 e 6	56,70	9

Fonte: Elaborado pelo autor (2018)

A seguir serão mostradas algumas configurações obtidas pelos algoritmos. A Figura 5 mostra os resultados obtidos com o algoritmo guloso para p=4 na configuração de distância em linha reta e distâncias reais em Vitória. A estação escolhida na configuração de distância em linha reta foi destacada com ícone verde e a estação escolhida para distâncias reais, em rosa, para melhor visualização espacial.

Além disso, temos ícones roxos que mostram estações escolhidas em comum nas duas configurações. Os ícones em laranja são as medianas (estações) não

escolhidas. A Figura 6 mostra os resultados obtidos com a metaheurística CRO para p=4 na configuração de distância em linha reta.

Figura 5 – Configuração de 4 estações escolhida com algoritmo guloso

Fonte: Elaborado pelo autor (2018)

Figura 6 – Configuração de 4 estações escolhida com metaheurística CRO

Fonte: Elaborado pelo autor (2018)

4.2 Base de Dados OR-LIBRARY

Para validar os algoritmos propostos, comparamos os resultados obtidos com os valores de uma base de dados da literatura: OR-Library (BEASLEY, 2017). Esta base possui 40 instâncias diferentes para problemas de p-mediana não capacitado.

Para obtenção dos resultados com os algoritmos propostos, o algoritmo guloso foi repetido uma única vez para cada um destes 40 casos, devido sua natureza determinística. Já a metaheurística CRO foi executada 15 vezes para cada um dos 40 casos, pois possui elementos aleatórios.

Nestes testes, o algoritmo guloso proposto obteve os mesmos resultados apresentados na base em 26 dos 40 problemas propostos.

Já a metaheurística CRO atingiu os resultados ótimos em 32 casos. Os maiores *gaps* encontrados pelo algoritmo guloso foram para pmed5 e pmed19, sendo de 0,14%, como mostrado na Tabela 8, que se encontra no apêndice A. Para o CRO, os maiores *gaps* foram em pmed15 e pmed20, sendo ambas de 0,06%. Lembrando que, como a metaheurística CRO é probabilística, também foi calculado o desvio médio para os 40 problemas, pois cada uma das 15 vezes que era executada a metaheurística, ela podia apresentar um resultado diferente.

Gap, nesse caso, é utilizado para especificar a diferença percentual entre o valor obtido na função objetivo pelos algoritmos e o valor ótimo da base da literatura. O *gap* mostrado para o algoritmo CRO é em relação ao melhor resultado obtido por ele. Na estatística, desvio médio é a medida da dispersão de um dado em relação a uma média, obtida por:

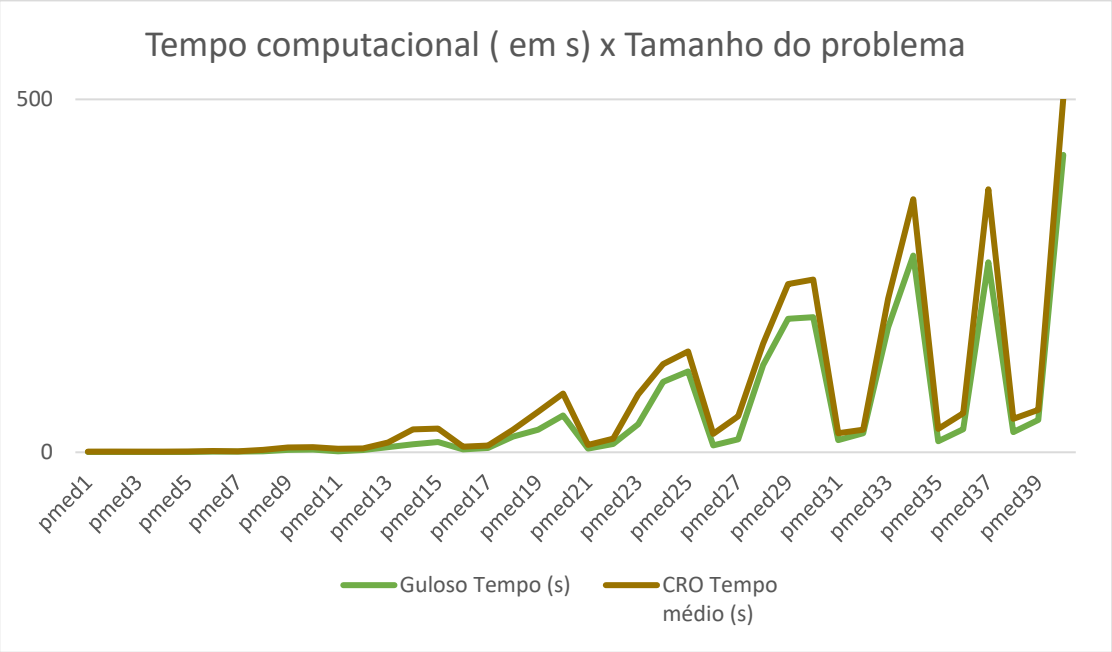
$$DM = \sum \frac{\left|X - \overline{X}\right|}{N} \qquad (11)$$

Como foi utilizado o mesmo computador para realizar os testes com o algoritmo Guloso e o CRO, foi feita a comparação do tempo computacional gasto. Para os problemas de menores instâncias, o algoritmo guloso foi, em média, 2 vezes mais rápido que o algoritmo CRO.

Para problemas com maiores instâncias, o algoritmo guloso continuou sendo mais rápido. De acordo com as literaturas consultadas, o algoritmo guloso sempre

teve um menor tempo computacional comparado a algumas metaheurísticas, como: Algoritmo Genético, *Iterated Local Search*, CRO. Então, o tempo computacional menor do guloso corresponde com o esperado e está mostrado na Figura 7.

Figura 7 – Tempo computacional x Tamanho do problema



Fonte: Elaborado pelo autor (2018)

Em relação ao desempenho, a metaheurística CRO mostrou-se melhor, pois atingiu o valor ótimo da base 32 vezes contra 26 vezes do algoritmo guloso, o que também era esperado.

Os resultados obtidos pelo algoritmo guloso e pelo CRO estão mostrados na Tabela 6, onde temos: o nome da instância, o valor da função objetivo, o tamanho da instância do problema, o número de medianas, os gaps (%), o tempo gasto (em s), e o desvio médio (somente para a metaheurística CRO).

4.3 Comparação dos Resultados com os da Literatura

Para maior validação dos algoritmos desenvolvidos, além da comparação feita anteriormente com os valores ótimos da base OR-Library, também foi estabelecida uma comparação com os valores encontrados por Daskin e Maass

(2015), que utilizaram a Relaxação Lagrangeana para resolução do problema de *p*-mediana para a mesma base.

Apesar da Relaxação Lagrangeana desenvolvida por Daskin e Maass (2015) ter alcançado o valor ótimo em 39 dos 40 casos, o algoritmo Guloso foi mais rápido em 24 dos 40 casos e com *gap* máximo de 0,14%. Fazendo uma comparação com o CRO, este também foi mais rápido em 16 dos 40 casos com *gap* máximo de 0,06% contra 19,5% da Relaxação Lagrangeana.

Além disso, pode ser feita uma comparação com o tempo computacional gasto para resolução do problema. Daskin e Maass (2015) relatam que o maior tempo gasto foi de 767,16s na instância *pmed36*.

Já os maiores tempos dos algoritmos desenvolvidos neste trabalho foram de 421,34s e 501,57s do algoritmo Guloso e do CRO, respectivamente.

Observamos que tanto o Algoritmo Guloso como o CRO obtiveram resultados satisfatórios, pois tiveram *gaps* menores comparados com o resultado encontrado por Daskin e Maass (2015). Além disso, o maior tempo computacional dos algoritmos desenvolvidos neste trabalho foi menor nessa comparação.

Esta comparação é aproximada visto que Daskin e Maass (2015) usaram um computador com 2.7GHz, Intel Core i7 e 1600 MHz DDR3. Lembrando que o computador utilizado foi uma máquina de 64 bits, processador Intel Core i5 de 2,4 GHz, 8GB de memória, que tem desempenho menor. O Intel Core i7 tem *average CPU mark* igual a 5188 e o i5 *average CPU mark* com 3282 (CPU Benchmarks, 2018).

5 CONSIDERAÇÕES FINAIS

Neste artigo, foi apresentado um estudo sobre localização de estações de carregamento de veículos elétricos numa região metropolitana do ES.

Para a solução deste problema foram propostos dois algoritmos: algoritmo guloso e um algoritmo baseado na metaheurística CRO. Para validar os algoritmos propostos foi utilizada a base OR-Library contendo 40 instâncias do problema de *p*-mediana não capacitado. Os resultados obtidos pelos dois algoritmos foram comparados com resultados ótimos da literatura dessa base. Os valores alcançados por esses dois algoritmos também foram comparados com resultados da literatura

utilizando outra estratégia de otimização através de um algoritmo com Relaxação Lagrangeana.

O objetivo do trabalho foi comparar qual algoritmo teria o melhor desempenho na localização das estações de carregamento. Os resultados mostraram que os dois algoritmos tiveram bons resultados tanto no teste com a base de dados *OR-Library* como no caso real da região metropolitana de Vitória-ES.

Comparando os dois algoritmos, foi possível concluir que o algoritmo guloso apresentou menor tempo computacional em relação ao CRO. Entretanto, a metaheurística CRO apresenta melhores resultados (maior qualidade) na resolução do problema. Os dois algoritmos se mostraram competitivos e aplicáveis a outros problemas de natureza combinatória.

Ficou evidente, também, que há divergência de resultados na consideração de distâncias em linha reta e rotas reais. Portanto, necessitamos escolher o modelo de distâncias reais, já que a diferença nos valores das funções objetivos é considerável. As distâncias reais representam melhor a realidade da distância percorrida pelos veículos.

Nessa região metropolitana considerada foram necessárias poucas estações, pois a região não é tão extensa territorialmente. A quantidade de estações é influenciada pela distância que pode ser percorrida sem que a bateria do VE necessite de recarga.

Podemos concluir, também, que o planejamento das estações de carregamento de VEs necessita de uma interferência na infraestrutura de uma região, levando a um melhor custo-eficácia, que é um dos alicerces para a evolução deste tipo de veículo.

Como trabalho futuro, pretendemos estabelecer custos diferenciados para as estações de carregamento, números diferentes de baias de recarga e ainda tipos diferentes de carregadores: rápidos, semirrápidos e carregamento em tempo normal. Assim, podemos estudar no futuro os impactos nos resultados dessas modificações nas localizações e nos números das estações de carregamento.

AGRADECIMENTOS

Os autores agradecem a Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) pelo apoio, dentro do Projeto PIBIC-IFES, Edital

PIBIC/PIBITI - 2016, Resolução nº 143/2016, Nível I (Ifes: projeto nº PJ00002933 e plano de trabalho nº PT00004475).

REFERÊNCIAS

BAI, R.; WOODWARD, J. R.; SUBRAMANIAN, N.; CARTLIDGE, J. Optimisation of transportation service network using κ-node large neighbourhood search. **Computers & Operations Research**, v. 89, p. 193-205, 2018. http://dx.doi.org/10.1016/j.cor.2017.06.008

BAOUCHE, F., BILLOT, R., El FAOUZI, N.-E e TRIGU. Efficient allocation of electric vehicles charging stations: optimization model and application to a dense urban network. **IEEE Intelligent Transportation Systems**, p. 33-43, Paris, 2014. http://dx.doi.org/10.1109/MITS.2014.2324023

BARBOSA, L. W. G; GAMARANO, C. G.; PEREIRA, A. L. C.; POLICARPO, R. V. S.; MAPA, S. M. S. A Pesquisa em Trade-Offs de Custos Logísticos: Estudo Bibliométrico no Período de 2006 a 2016. **Revista Produção Online**, v.18, n. 2, p. 641-664, 2018. http://dx.doi.org/10.14488/1676-1901.v18i2.2882

BASTOS, E. A. **Otimizaçao de Seções Retangulares de Concreto Armado Submetidas à Flexo-Compressao Oblíqua Utilizando Algoritmos Genéticos.** 2004. 168 f. Dissertação (Mestrado em Engenharia Civil). Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

BEASLEY, J. E. **OR-LIBRARY**. Disponível em: http://people.brunel.ac.uk/~mastjjb/jeb/info.html. Acesso em: 20 set 2017.

BNEF - Blombeerg New Energy Finance. Disponível em: https://about.bnef.com/. Acesso em: 20 jun. 2017.

CHIYOSHI, F., GALVÃO, R. D. A statistical analysis of simulated annealing applied to the *p*-median problem. **Annals of Operation Research**, v. 96, n. 4, p. 61-74, 2000. http://dx.doi.org/10.1023/A:1018982914742

CHRISTOFIDES, N.; BEASLEY, J. E. A tree search algorithm for the p-median problem. **European Journal of Operational Research**, v.10, p. 196 – 204, 1982. https://doi.org/10.1016/0377-2217(82)90160-6

COSTA, C. E. S. **Aplicação de técnicas de pesquisa operacional na determinação de setores de atendimento de uma concessionária de energia.** 2005. 146 f. Dissertação (Mestrado em Métodos Numéricos em Engenharia).Universidade Federal do Paraná, Curitiba, 2005.

CPU Benchmarks. Disponível em: https://www.cpubenchmark.net/cpu_list.php. Acesso em: 23 jul. 2018.

DASKIN, M. S.; MAASS, K. L. **Chapter 2 - The p-Median Problem.** Springer International Publisinhg Switzerland. 26 f. Suíça, 2015. http://dx.doi.org/DOI 10.1007/978-3-319-13111-5_2

DENATRAN – Departamento Nacional de Trânsito. Disponível em: http://www.denatran.gov.br/estatistica/257-frota-2015. Acesso em:16 abr. 2017.

EDP Portugal. Disponível em: https://www.edp.pt/pt/sustentabilidade/ied/Pages/veiculoseletricos.aspx. Acesso em: 02 abr. 2017

FEO, A. T.; RESENDE, M., Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, n. 2, p. 109-133. 1995. http://dx.doi.org/10.1007/BF01096763

FUCHIGAMI, H. Y.; MOURA, M. A. S.; BRANCO, F. J. C. Modelos Matemáticos para Programação de Job Shop com Tempos de Setup Independentes da Sequência. **Revista Produção Online**, v.17, n. 1, p. 245-267, 2017. http://dx.doi.org/10.14488/1676-1901.v17i1.2504

GOOGLE MAPS. Disponível em: https://www.google.com.br/maps. Acesso em: 20 fev. 2017.

GOMES, F. M.; PEREIRA, F. M.; MARINS, F. A. S.; SILVA, M. B. Estudo comparativo entre os métodos gradiente reduzido generalizado e algoritmo genético em otimização com múltiplas respostas. **Revista Produção Online**, v.17, n. 2, p. 592-619, 2017. http://dx.doi.org/10.14488/1676-1901.v17i2.2566

HANSEN, P.; MLADENOVIC, N.; PEREZ-BRITOS, D. Variable neighborhood decomposition search. **Journal of Heuristics**. v. 7, n. 4, p. 335-350, 2001. http://dx.doi.org/10.1023/A:1011336210885

HÖRNER, D. **Resolução do problema das p-medianas não capacitado:** comparação de algumas técnicas heurísticas. 2009. 104 f. Dissertação. (Mestrado em Engenharia de Produção). Universidade Federal de Santa Catarina, Florianópolis, 2009.

LAM, A. Y. S., Leung Y-W., Chu, X. Electric Vehicle Charging Station Placement: Formulation, Complexity, and Solutions. **IEEE Transactions on Smart Grid,** v. 5, n. 6, p. 2846-2856, nov, 2014. http://dx.doi.org/10.1109/TSG.2014.2344684

LAM, A. Y. S.; LI, V. O. K. Chemical-reaction-inspired metaheuristic for optimization. **IEEE Transactions on Evolucionary Computacion**, v. 14, n. 3, p. 391-399, June, 2010. http://dx.doi.org/10.1109/TEVC.2009.2033580

MESTRIA, M. A Hybrid heuristic algorithm for the clustered traveling salesman problem. **Pesquisa Operacional**, Rio de Janeiro, v. 36, n. 1, p. 113-132, 2016. http://dx.doi.org/10.1590/0101-7438.2016.036.01.0113.

OLIVEIRA, M. G. **Sistema de localização de facilidades:** uma abordagem para mensuração de pontos de demanda e localização de facilidades. 2012. 91f. Dissertação (Mestrado em Ciências da Computação).Universidade Federal de Goiás, Goiás, 2012.

OSORIO, V. A. G. **Carregamento ótimo de veículos elétricos considerando as restrições das redes de distribuição de média tensão.** 2013. Dissertação (Mestrado em Engenharia Elétrica). Universidade Estadual Paulista, São Paulo, 2013.

RIECHI, J. L.; TORMOS, B.; HILLEBRAND, M. V. J. Otimização dos custos de frota urbana com uso de modelo combinado de life cycle cost e simulação de Monte Carlo. **Revista**

Produção Online, v.17, n. 2, p. 667-691, 2017. http://dx.doi.org/10.14488/1676-1901.v17i2.2627

ROCHA, A; DORINI, L. B. **Algoritmos gulosos:** definições e aplicações. Campinas, SP, 53 pp, 2004.

SILVA NETO, C. A.; SOUZA, J. C. S.; SCHILLING, M. T.; SANTOS, M. G. Melhoria da Segurança Dinâmica baseada em análise estocástica e metaheurística**. Revista Controle e Automação**. V. 23, n. 2, p. 216-230. 2012. Disponível em: http://www.scielo.br/pdf/ca/v23n2/v23n2a08. Acesso em: 08 set 2017. http://dx.doi.org/10.1590/S0103-17592012000200008

SOUZA, M. J. F. **Notas de aula**. Universidade Feral de Ouro Preto, Departamento de Computação, Ouro Preto, MG, 2008. Apostila. Disponível em: http://www.inf.ufpr.br/aurora/disciplinas/topicosia/tabu/InteligenciaComputacional.pdf. Acesso em: 13 nov. 2016.

STEINER, M. T. A. **Notas de aula**. Universidade Federal do Paraná, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, Curitiba, PR, 2003. Apostila.

TRAGANTALERNGSAK, S.; HOLT, J.; RÖNNQVIST, M. An exact method for the twoechelon, single-source, capacited facility location problem. **European Journal of Operational Research,** Amsterdam, v. 123, p. 473-489, 1999. http://dx.doi.org/10.1016/S0377-2217(99)00105-8

UOL – UNIVERSO ONLINE. São Paulo, Outubro, 2018. Disponível em: https://carros.uol.com.br/noticias/redacao/2018/10/19/bndes-libera-r-67-milhoes-para-fazer-rede-de-recarga-de-carros-eletricos.htm. Acesso em: 19 out 2018.

Artigo recebido em: 23/07/2018 e aceito para publicação em: 20/01/2019
DOI: http://dx.doi.org/10.14488/1676-1901.v19i1.3324

APÊNDICE A – **Resultados da resolução dos 40 problemas da base OR- Library**

Tabela 8 – Resultados da resolução dos 40 problemas da base OR-Library (continua)

	OR Library			Guloso			CRO			
Nome do arquivo	**FO**	**Tamanho**	**p**	**FO**	**Gap(%)**	**Tempo (s)**	**FO**	**Gap (%)**	**Desvio médio (%)**	**Tempo médio (s)**
pmed1	5819	100	5	5819	0,00	0,46	5819	0,00	0,00	0,98
pmed2	4093	100	10	4093	0,00	0,49	4093	0,00	0,03	0,97
pmed3	4250	100	10	4253	0,07	0,49	4253	0,07	0,07	0,98
pmed4	3034	100	20	3034	0,00	0,49	3033	0,03	0,03	0,99
pmed5	1355	100	33	1356	0,14	0,54	1355	0,00	0,11	1,10
pmed6	7824	200	5	7824	0,00	0,93	7824	0,00	0,00	1,78
pmed7	5631	200	10	5631	0,00	0,95	5631	0,00	0,00	1,37
pmed8	4445	200	20	4445	0,00	1,40	4445	0,00	0,39	3,46
pmed9	2734	200	40	2734	0,00	3,47	2734	0,00	0,13	6,74
pmed10	1255	200	67	1256	0,08	3,88	1255	0,00	0,00	7,23
pmed11	7696	300	5	7696	0,00	1,36	7696	0,00	0,04	4,91
pmed12	6634	300	10	6634	0,00	3,11	6634	0,00	0,02	5,37
pmed13	4374	300	30	4374	0,00	7,37	4374	0,00	0,06	13,78
pmed14	2968	300	60	2970	0,07	11,45	2969	0,03	0,00	32,45
pmed15	1729	300	100	1731	0,12	14,48	1730	0,06	0,09	33,56
pmed16	8162	400	5	8162	0,00	4,05	8162	0,00	0,01	7,89
pmed17	6999	400	10	6999	0,00	6,11	6999	0,00	0,00	9,54
pmed18	4809	400	40	4809	0,00	22,13	4809	0,00	0,00	31,93
pmed19	2845	400	80	2849	0,14	32,03	2845	0,00	0,24	57,40
pmed20	1789	400	133	1790	0,06	52,18	1790	0,06	0,11	83,27
pmed21	9138	500	5	9138	0,00	5,14	9138	0,00	0,03	10,36
pmed22	8579	500	10	8579	0,00	11,78	8579	0,00	0,03	19,18
pmed23	4619	500	50	4620	0,02	39,65	4619	0,00	0,00	81,88
pmed24	2961	500	100	2962	0,03	99,79	2962	0,03	0,06	125,02
pmed25	1828	500	167	1830	0,11	114,56	1829	0,05	0,00	142,86
pmed26	9917	600	5	9917	0,00	9,86	9917	0,00	0,00	26,14
pmed27	8307	600	10	8307	0,00	18,27	8307	0,00	0,07	51,08
pmed28	4498	600	60	4498	0,00	123,71	4498	0,00	0,00	154,09
pmed29	3033	600	120	3033	0,00	189,21	3033	0,00	0,11	238,34
pmed30	1989	600	200	1990	0,05	191,34	1989	0,00	0,45	244,67
pmed31	10086	700	5	10086	0,00	17,23	10086	0,00	0,07	26,98
pmed32	9297	700	10	9297	0,00	27,06	9297	0,00	0,00	31,96
pmed33	4700	700	70	4703	0,06	177,08	4701	0,02	0,02	217,71
pmed34	3013	700	140	3013	0,00	278,74	3013	0,00	0,00	358,54
pmed35	10400	800	5	10400	0,00	15,67	10400	0,00	0,09	33,10
pmed36	9934	800	10	9934	0,00	32,65	9934	0,00	0,12	55,27
pmed37	5057	800	80	5064	0,14	269,08	5057	0,00	0,02	372,69

Tabela 8 – Resultados da resolução dos 40 problemas da base OR-Library (conclusão)

Nome do arquivo	OR Library			Guloso			CRO			
	FO	Tamanho	p	FO	Gap(%)	Tempo (s)	FO	Gap (%)	Desvio médio (%)	Tempo médio (s)
pmed38	11060	900	5	11060	0,00	28,56	11060	0,00	0,00	47,37
pmed39	9423	900	10	9423	0,00	45,89	9423	0,00	0,00	60,02
pmed40	5128	900	90	5131	0,06	421,34	5130	0,04	0,04	501,57

Fonte: Elaborado pelo autor (2018)

APÊNDICE B - **Exemplo de cálculo de distância em linha reta usando latitude e longitude**

Para exemplo, foi feito o cálculo utilizando as latitudes e longitudes do local candidato 1 (Ok Hipermercado) e cliente 6 (Figura 8). As latitudes e longitudes deles estão mostradas nas Tabelas 2 e 3.

Quadro 1 – Exemplo de cálculo de distância em linha reta

1º: Transformação de graus decimais do local candidato em graus, minutos e segundos:

Latitude:
20,29359 – Separe a parte inteira. Então, temos 20º.
Depois: 20,29359 – 20 = 0,29359
0,29359 * 60 = 17,6154 – Separe a parte inteira. Temos 17'.
Após: 17,6154 – 17 = 0,6154
0,6154 * 60 = 36,924. Então, temos 36,924''.
Assim: 20º 17' 36,924''

Longitude:
40,30146 - Separe a parte inteira. Então, temos 40º.
Depois: 40,30146 – 40 = 0,30146
0,30146 * 60 = 18,0876 – Separe a parte inteira. Temos 18'.
Após: 18,0876 – 18 = 0,0876
0,0876 * 60 = 5,256. Então, temos 5,256''.
Assim: 40º 18' 5,256''

2º: Transformação de graus decimais do cliente em graus, minutos e segundos:
Latitude:
20,27945– Separe a parte inteira. Então, temos 20º.
Depois: 20,27945 – 20 = 0,27945
0,27945 * 60 = 16,767 – Separe a parte inteira. Temos 16'.
Após: 16,767 – 16 = 0,767
0,767 * 60 = 46,02. Então, temos 46,02''.
Assim: 20º 16' 46,02''

Longitude:
40,29338 - Separe a parte inteira. Então, temos 40º.
Depois: 40,29338 – 40 = 0,29338
0,29338 * 60 = 17,6028 – Separe a parte inteira. Temos 17'.
Após: 17,6028 – 17 = 0,6028
0,6028 * 60 = 36,168. Então, temos 36,168''.
Assim: 40º 17' 36,168''

3º: Cálculo de DLA (graus):
$DLA_G = [(-20) – (-20)] * 60 = 0$
$DLA_M = [(-17) – (-16)] * 1 = -1$
$DLA_S = [(-36,924) – (-46,02)]/60 = 0,1516$

$DLA_{total} = DLA_G + DLA_M + DLA_S = -0,8484$

4º: Transformação para distância com milha:
$| DLA_{total} | * 1852 = 1571,23$ m

5º: Cálculo de DLO (graus):
DLO_G = [(-40) – (-40)] * 60 = 0
DLO_M = [(-18) – (-17)] * 1 = -1
DLO_S = [(-5,256) – (-36,168)]/60 = 0,5152

$DLO_{total} = DLO_G + DLO_M + DLO_S$ = -0,4848

6º: Transformação para distância com milha:

$| DLA_{total} |$ * 1852 = 897,85 m

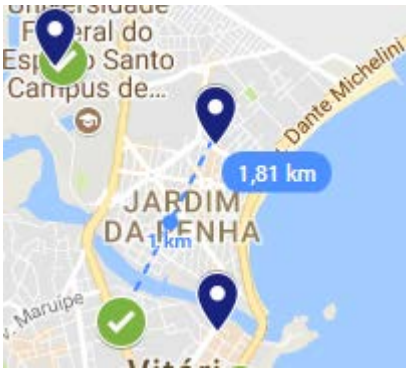
7º: Distância entre dois pontos:

$$DE_{AB} = [(1571,23)^2 + (897,85)^2]^{1/2}$$

$$DE_{AB} = 1809,67\ m$$

Fonte: Elaborado pelo autor (2018)

Figura 8 – Distância entre local candidato 1 e cliente 6 no Google Maps



Fonte: Elaborado pelo autor (2018)