INSTITUTO FEDERAL DO ESPÍRITO SANTO
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIAS SUSTENTÁVEIS
MESTRADO PROFISSIONAL EM TECNOLOGIAS SUSTENTÁVEIS

DANILO CESAR AZEREDO SILVA

LOCALIZAÇÃO DE ESTAÇÕES DE CARREGAMENTO DE VEÍCULOS ELÉTRICOS UTILIZANDO HEURÍSTICAS E PADRÕES REAIS DE DESLOCAMENTO VEICULAR

Vitória
2018

DANILO CESAR AZEREDO SILVA

LOCALIZAÇÃO DE ESTAÇÕES DE CARREGAMENTO DE VEÍCULOS ELÉTRICOS UTILIZANDO HEURÍSTICAS E PADRÕES REAIS DE DESLOCAMENTO VEICULAR

Dissertação apresentada ao Programa de Pós-Graduação em Tecnologias Sustentáveis do Instituto Federal do Espírito Santo como requisito parcial para obtenção do Título de Mestre em Tecnologias Sustentáveis.

Orientador: Prof. Dr. Mário Mestria

Vitória

2018

(Biblioteca Nilo Peçanha do Instituto Federal do Espírito Santo)

S586l Silva, Danilo Cesar Azeredo.

Localização de estações de carregamento de veículos elétricos utilizando heurísticas e padrões reais de deslocamento veicular / Danilo Cesar Azeredo Silva. – 2018.

185 f. : il. ; 30 cm

Orientador: Mário Mestria.

Dissertação (mestrado) – Instituto Federal do Espírito Santo, Programa de Pós-graduação em Tecnologias Sustentáveis, Vitória, 2018.

1. Veículos elétricos . 2. Baterias elétricas. 3. Eletricidade nos transportes. 4. Sustentabilidade. I. Silva, Danilo Cesar Azeredo. II. Instituto Federal do Espírito Santo. III. Título.

CDD: 629.2293

DANILO CESAR AZEREDO SILVA

LOCALIZAÇÃO DE ESTAÇÕES DE CARREGAMENTO DE VEÍCULOS ELÉTRICOS UTILIZANDO HEURÍSTICAS E PADRÕES REAIS DE DESLOCAMENTO VEICULAR

Dissertação apresentada ao Programa de Pós-Graduação em Tecnologias Sustentáveis do Instituto Federal do Espírito Santo como requisito parcial para obtenção do Título de Mestre em Tecnologias Sustentáveis.

Aprovado em 27 de abril de 2018.

COMISSÃO EXAMINADORA

Prof. Dr. Mário Mestria
Instituto Federal do Espírito Santo
Orientador

Prof. Dr. Luciano Lessa Lorenzoni
Instituto Federal do Espírito Santo
Membro Interno

Prof. Dr. Rodrigo Varejão Andreão
Instituto Federal do Espírito Santo
Membro Interno

Prof. Dr. Oscar Luiz Teixeira de Rezende
Instituto Federal do Espírito Santo
Membro Interno

Prof. Dr. André Renato Sales Amaral
Universidade Federal do Espírito Santo
Membro Externo

Para minha esposa, Denisa, por sempre acreditar em mim e me apoiar.
Para meu filho, Patrick, por me dar uma razão para seguir a minha jornada terrena.

AGRADECIMENTOS

A Deus por ter me criado e me permitido concluir mais uma prova da minha jornada.

A minha esposa, Denisa, por sempre acreditar em mim e me apoiar.

Ao meu filho, Patrick, por me dar uma razão para seguir a minha jornada terrena.

Ao meu orientador, Prof. Dr. Mário Mestria, pelos ensinamentos, orientações, paciência e disponibilidade.

Ao Prof. Dr. Rodrigo Varejão Andreão, pelos ensinamentos e por acreditar em mim.

A todos os professores do Instituto Federal do Espírito Santo que contribuíram para o trabalho aqui finalizado.

A primeira turma do Mestrado em Tecnologias Sustentáveis do Instituto Federal do Espírito Santo, 2016, da qual faço parte.

RESUMO

Para melhorar a sustentabilidade ambiental, muitos países vão incluir a eletrificação de parte de seus sistemas de transporte, nos planos futuros de cidades inteligentes. Portanto, o número de veículos elétricos (EVs) circulando em cidades poderá crescer de forma significativa. Embora existam diversas maneiras de recarregar baterias de EVs, estações de carregamento comerciais deverão ser a sua principal fonte de energia de recarga. Desse modo, a introdução de EVs em grandes centros urbanos dependerá da implantação otimizada de estações de carregamento comerciais, em termos de custo e localização. Nesse trabalho um problema matemático de otimização foi utilizado para criar uma ferramenta de suporte à decisão, com o objetivo de localizar estações de carregamento comerciais de EVs, de forma a minimizar seu custo de implantação, bem como os custos de deslocamento dos usuários de EVs às mesmas. Um conjunto inicial de locais candidatos a receberem estações de carregamento, dentro de uma cidade metropolitana, foi determinado utilizando-se sistemas de mapeamento disponíveis na Internet. A demanda de recarga foi representada como agrupamentos de EVs localizados em pontos selecionados da mesma cidade. Adicionalmente, foram utilizados padrões reais de deslocamento entre os agrupamentos de EVs e estações de carregamento, com rotas obtidas de sistemas de navegação disponíveis na Internet, o que permitiu uma representação mais precisa das necessidades de consumo de energia dos EVs. Para a resolução do problema de localização foram utilizados um otimizador comercial de programação linear e inteira, além de uma adaptação da metaheurística *Chemical Reaction Optimization* (CRO). Os resultados obtidos poderão auxiliar a futura implantação de estações de carregamento em áreas metropolitanas.

Palavras-chave: Veículos Elétricos. Estações de carregamento. CRO. Metaheurísticas. Logística e Transportes.

ABSTRACT

To improve environmental sustainability, many countries will include electrification of part of their transportation systems in their plans for smart cities. Therefore, the number of electric vehicles (EVs) circulating in cities may grow significantly. Although there are several different ways to recharge EV batteries, commercial charging stations should become their primary source of charging power. Therefore, the introduction of EVs in large urban centers will depend on the optimized deployment of commercial charging stations in terms of cost and location. In this work, a mathematical optimization problem was used to create a decision support tool to locate commercial EV charging stations in order to minimize deployment costs, as well as EV owners´ costs to drive to nearest charging stations, for recharging. An initial set of candidate sites to have charging stations installed, in a metropolitan city, was determined using mapping systems available on the Internet. The recharging demand was represented as EV clusters located in the same city. Moreover, real driving distances from EV clusters to charging stations, gathered from navigation systems available on the Internet, were used. That allowed for a more precise representation of EVs energy consumption requirements. To solve the location problem, a linear and integer programming commercial optimizer was used, as well as an adaptation of the Chemical Reaction Optimization (CRO) metaheuristic. Results could help future deployment of charging stations in metropolitan areas.

Keywords: Electric Vehicles. Charging Stations. Chemical Reaction Optimization. Metaheuristic. Logistics and Transportation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Colab. – Colaboradores

Dist. – Distância

Est. – Estação

et al. – Et alii (e outros)

Exec. – Execuções

Impr. - *Improvement*

Inst. – Instância

Iter. – Iterações

Inter. – Intermolecular(es)

Max. – Máximo

Med. – Médio (a)

Min. – Mínimo

Num. – Número

Obj. – Objetivo

Sim. – *Simulated*

Sint. – Síntese

Tam. – Tamanho

w/o – *Without*

LISTA DE SIGLAS

API – *Application Programming Interface*

CLP – Conjunto de Listas de Proximidade

CPFL – Companhia Paulista de Força e Luz

CPLEX – Programa otimizador linear e inteiro, de autoria do fabricante IBM

CPMP – *Capacitated p-Median Problem*

CPU – *Central Processing Unit* (Unidade Central de Processamento)

CRO – *Chemical Reaction Optimization*

CS – *Clustering search*

CSV – *Comma Separated Values*

DPX – *Distance Preserving Crossover*

EV – *Electric Vehicle*

EVSE – *Electric Vehicle Supply Equipment*

FCLP – *Fixed Charge Location Problem*

FI – *Fast Interchange*

GA – *Genetic Algorithm*

Gap – Diferença percentual entre o valor da função objetivo obtido pelo método heurístico, exato ou híbrido e o seu respectivo valor ótimo (ou melhor valor conhecido), na resolução de um problema

GRAMPS – *Greedy Random Adaptive Memory Programming Search*

HEV – *Hybrid Electric Vehicle*

HTTP – *Hypertext Transfer Protocol*

ICE – *Internal Combustion Engine*

ID - Identificação

IRMA – *Iterated Reduction Matheuristic Algorithm*

KE – *Kinetic Energy* (Energia Cinética)

LP – Lista de Proximidade

LSCP – *Location Set Covering Problem*

MALP – *Maximum Availability Location Problem*

MCLP – *Maximal Covering Location Problem*

MIP – *Mixed Integer Programming*

NS – *Neighborhood Search*

OPL – *Optimization Programming Language*

PC – *Personal Computer*

PE – *Potential Energy* (Energia Potencial)

PEV – *Plug-in Electric Vehicle*

PHEV - *Plug-in Hybrid Electric Vehicle*

RAM – *Random Access Memory*

RL – Relaxação Lagrangiana

SA – *Simulated Annealing*

SAE – *Society of Automotive Engineers*

SS – *Scatter Search*

Tam. – Tamanho

TMEXCLP – *Time dependent Maximal Expected Covering Location Problem*

TS – *Tabu Search* (Busca Tabu)

Vix – Base de dados construída a partir de locais da região da Grande Vitória, ES

VNS – *Variable Neighborhood Search*

XML – *Extended Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A melhoria da qualidade do ar nas grandes áreas metropolitanas dependerá, em grande parte, da redução de emissões produzidas por veículos de combustão interna. Os mecanismos antipoluição e de redução de emissões utilizados atualmente nos veículos propulsionados por derivados de petróleo ou combustíveis de origem vegetal parecem ter chegado ao seu ápice de desenvolvimento, não sendo esperados ganhos significativos para o futuro. Por outro lado, a frota mundial de veículos vem aumentando de forma alarmante a cada ano, inclusive no Brasil, onde somente a frota de automóveis saltou de 28 milhões para 51 milhões de veículos, entre 2006 e 2016 (DEPARTAMENTO NACIONAL DE TRÂNSITO, 2016).

Uma forma mais eficiente para atingir a redução de emissões é a da intensificação da circulação de veículos propulsionados por energias alternativas, como a elétrica. Adicionalmente, a inserção de veículos elétricos (EVs, do inglês *Electric Vehicles*), na forma de automóveis, ônibus ou motocicletas, pode contribuir significativamente no alívio da dependência do petróleo e na redução da emissão de gases causadores do efeito estufa.

No Brasil, em 2015, 65,6% da energia elétrica produzida foi de origem hidráulica e eólica (EMPRESA DE PESQUISA ENERGÉTICA, 2016), constituindo-se em fontes de energia renovável. Esta característica da matriz energética brasileira torna ainda mais atraente a eletrificação de nossos sistemas de transporte, uma vez que a necessidade do uso de geração termoelétrica ou nuclear é significativamente menor que em muitos outros países. Considere-se ainda a localização geográfica do Brasil, que favorece a geração de energia solar e eólica e que poderá ser usada, no futuro, para suprir parte do aumento de demanda energética causada por uma frota crescente de EVs.

Promover o uso de EVs é, portanto, uma solução tecnológica sustentável, de longo prazo, para manter um balanço saudável entre mobilidade, consumo de energia e controle da qualidade do ar em áreas urbanas. Sendo assim, a fim de melhorar a sustentabilidade ambiental, muitos países vão incluir a eletrificação de parte de seus sistemas de transporte nos planos futuros de cidades inteligentes (*Smart Cities*) (LAM et al., 2014). Portanto, o número de EVs circulando em cidades poderá crescer de forma significativa. Embora existam diversas maneiras de recarregar baterias de EVs, estações de carregamento comerciais deverão ser a principal fonte de energia de recarga. Desse modo, a introdução de EVs em grandes centros urbanos dependerá da implantação de estações de carregamento, otimizadas em termos de custo e localização.

A localização eficiente de estações de carregamento comerciais de EVs é um componente crítico no projeto das cidades inteligentes (*Smart Cities*), uma vez que tais estações deverão ser a principal fonte de energia de carregamento de EVs em áreas metropolitanas, devido às dificuldades de carregamento residencial (WU et al., 2010). Se as estações não estiverem presentes em número suficiente e convenientemente próximas aos potenciais consumidores, ou seja, os futuros proprietários de EVs, a adoção da tecnologia poderá ser desestimulada. Além disso, as estações deverão estar suficientemente dispersas, de modo que donos de EVs possam circular sem restrições por toda a cidade (LAM et al., 2013). Do ponto de vista dos futuros proprietários das estações, o seu custo de implantação e operação deverá ser minimizado, a fim de tornarem-se viáveis e atrativas financeiramente. Por fim, locais candidatos a se tornarem estações deverão respeitar restrições físicas, tecnológicas, humanísticas, econômicas, de recursos naturais, bem como a legislação vigente (PLUGINSITES, 2017).

Tradicionalmente, estações de carregamento são colocadas em locais selecionados aleatoriamente, à medida em que se tornem necessárias. No entanto, esses locais podem não ser os mais adequados para o tráfego de EVs e/ou necessidades de carregamento, onde uma recarga pode, normalmente, levar várias horas. Essa infraestrutura de carregamento de EVs atual tem sido, portanto, um impedimento à adoção generalizada de EVs (DAI et al., 2015). No passado, as distâncias de viagem até as estações de carregamento, quando disponíveis, contribuíram para que os EVs não se tornassem populares. Mais recentemente, devido ao desenvolvimento de baterias de maior capacidade e o aumento da eficiência e capacidade dos motores elétricos, a adoção dos EVs tem crescido de forma constante.

Um EV típico opera com uma grande célula de armazenamento de energia ou bateria recarregável. Portanto, a distância que os EVs podem viajar com uma única carga é limitada pela capacidade da bateria. Apesar dos recentes avanços na tecnologia de baterias para uso em EVs, a capacidade de armazenamento das mesmas ainda é relativamente baixa e seus tempos de recarga, altos (HANNAN et al., 2014). Assim, a autonomia dos EVs tem se mantido relativamente curta. Num dado momento, a autonomia remanescente de um EV depende, tipicamente, da distância já percorrida pelo mesmo, da topologia rodoviária, das condições de tráfego e até mesmo das condições meteorológicas (BAOUCHE et al., 2014).

Nos EUA, a maioria dos proprietários de EVs têm capacidade de carregamento em casa suficiente para viagens locais de curto alcance. No entanto, tais proprietários frequentemente encontram-se em situações nas quais os veículos precisam ser reabastecidos fora de casa, como

em locais de trabalho, centros comercias ou locais de eventos. Segundo Morrow e colaboradores (2008), no relatório do Departamento de Energia dos EUA, intitulado "*Plug-in Hybrid Electric Vehicle Charging Infrastructure Review*", observa-se que, quando estações de carregamento comerciais estão disponíveis fora das residências, a faixa média de perda de carga dos EVs, medida em termos de autonomia, cai de 40 para 13 milhas.

Um estudo recente da FGV Energia mostra que a frota mundial de elétricos e híbridos no ano passado era de 2 milhões de veículos para passageiros, excluindo-se ônibus e motocicletas. A previsão é que até 2020 a frota chegue a 13 milhões e, em 2030, a 140 milhões, ou 10% da frota total de carros (REVISTA PEQUENAS EMPRESAS GRANDES NEGÓCIOS, 2017). A Navigant Research (2016) estima que as vendas anuais de EVs nos EUA aumentarão para 186.000 em 2018 e a quantidade de EVs em circulação no mundo chegará a 37 milhões em 2025. O governo da China estabeleceu metas para ter cerca de 1 milhão de EVs em circulação até o fim de 2018. No entanto, a fim manter a tendência de crescimento apontada por estas estatísticas, requerer-se-á uma infraestrutura de carregamento de EVs adequada e com capacidade geograficamente distribuída. Infelizmente, mesmo em países que possuem frotas significativas de EVs, a infraestrutura atual compõe-se de relativamente poucas estações de carregamento comerciais e com localização esparsa. Ainda assim, na Alemanha, que possui uma frota de 24 mil veículos com emissão zero, já existem 2,8 mil postos de reabastecimento. Nos Estados Unidos, onde a frota elétrica chega a 275 mil unidades, existem 21,8 mil estações (ASSOCIAÇÃO BRASILEIRA DO VEÍCULO ELÉTRICO, 2016).

No Brasil, a utilização de EVs ainda é de pequena escala. Desde 2011 foram vendidos 5,9 mil carros elétricos e híbridos. No entanto a taxa de crescimento é alta. Em 2017, foram 2.079 veículos vendidos, ou quase o dobro de 2016. Este número representa cerca de 0,3% das vendas totais de veículos no Brasil. A quase totalidade das vendas é de híbridos, uma vez que os mesmos não necessitam de estações de carregamento. Sozinho, o híbrido Toyota *Prius*, que custa cerca de R$ 120 mil, respondeu por quase 80% das vendas de 2017, com 1.635 unidades. A Empresa de Pesquisa Energética (EPE) calcula que, em dez anos, 2,5% das vendas de carros no País serão de híbridos, o que equivalerá a 0,4% da frota total circulante (REVISTA PEQUENAS EMPRESAS GRANDES NEGÓCIOS, 2017).

Além da ausência de infraestrutura de carregamento adequada, a alta carga tributária e a ausência de subsídios para aquisição também se constituem em grandes entraves à popularização dos EVs, sejam puros ou híbridos. No entanto, a redução no imposto de

importação, de 35% para 7%, além de outros estímulos governamentais na área fiscal, a serem introduzidos em breve, deverão pavimentar o caminho para utilização desse tipo de veículo no Brasil. Adicionalmente, são esperados incentivos para a nacionalização de componentes básicos, como baterias e motores elétricos. Existem também iniciativas isoladas em cidades e Estados: o município de São Paulo concedeu isenção de IPVA e de rodízio, o qual restringe a circulação de veículos à combustão no centro da capital.

Na área de infraestrutura, uma parceria entre a Companhia Paulista de Força e Luz (CPFL) e a rede de postos de serviço Graal definiu a instalação de dois eletropostos de recarga de carros elétricos em rodovias (ASSOCIAÇÃO BRASILEIRA DO VEÍCULO ELÉTRICO, 2016). Atualmente a CPFL possui um total de 25 eletropostos, sendo dez públicos e os demais em parceria com empresas, mas poucos são de recarga rápida. Ainda é aguardada a regulamentação da venda de energia para essa finalidade pela Agência Nacional de Energia Elétrica (ANEEL). Hoje, os postos não podem cobrar pela recarga. Segundo a agência, a proposta de regulamentação será apreciada no primeiro semestre de 2018. A CPFL projeta que o Brasil terá ao menos 15 mil pontos de recarga elétrica até 2030 (REVISTA PEQUENAS EMPRESAS GRANDES NEGÓCIOS, 2017).

Segundo a revista Pequenas Empresas e Grandes Negócios, a empresa de tecnologia ABB negocia a instalação de vários equipamentos de carga rápida para baterias com uma rede de postos de combustível, shoppings, estacionamentos e aeroportos.

Apesar das dificuldades para comercialização de EVs nos país, em 2016 existiam seis modelos à venda, todos do tipo híbrido. Além disso, a montadora japonesa Nissan, anunciou que, até 2019, pretende fabricar o modelo elétrico puro Nissan *Leaf*, na sua fábrica de Resende, RJ. A previsão é de que o modelo estará completamente nacionalizado em 2020 (ESTADÃO, 2016).

Prevê-se que as dificuldades de carregamento residencial serão ainda mais acentuadas nas regiões metropolitanas do Brasil, uma vez que a maioria dos seus habitantes reside em prédios de apartamentos, que não possuem infraestrutura adequada para carregamento de EVs em suas garagens. Isso faz com que a existência de uma infraestrutura de estações de carregamento se torne ainda mais crucial para a adoção de EVs no Brasil.

Como vem ocorrendo em todo mundo, a eletrificação de parte da frota de veículos no Brasil, embora lenta, será inevitável. Deste modo, uma infraestrutura adequada de carregamento deverá ser desenvolvida para suportar o crescimento da frota.

Nesta dissertação de Mestrado, uma ferramenta de suporte à decisão para localização de estações de carregamento de EVs foi desenvolvida, com o objetivo de minimizar tanto os custos de implantação das estações de carregamento de EVs quanto os custos de deslocamento dos usuários de EVs às estações mais próximas aos mesmos. A ferramenta toma como entrada um conjunto inicial de locais candidatos a tornarem-se estações de carregamento, bem como um conjunto de centros de consumo. Também são fornecidos uma distância mínima de separação entre estações de carregamento, o custo de implantação de cada estação, as suas capacidades máximas de fornecimento de energia, a demanda de carregamento prevista para cada centro de consumo, a distância dos centros de consumo às estações candidatas e, finalmente, a energia dispendida por EVs típicos para se deslocarem dos centros de consumo às estações candidatas. Como base nos dados fornecidos, a ferramenta determina, do conjunto de estações candidatas, um subconjunto de estações cuja localização é ótima ou próxima da localização ótima.

Para determinar a localização das estações, utilizou-se um modelo matemático de otimização, originalmente proposto por Baouche e colaboradores (2014), para localização eficiente de estações de carregamento na cidade de Rhone, França. O modelo de otimização foi formulado como um problema de programação inteira e resolvido por métodos matemáticos exatos e heurísticos. A resolução por métodos matemáticos exatos, possibilitou a obtenção dos valores ótimos para as instâncias propostas, porém a um custo computacional relativamente alto, para as instâncias testadas. A utilização de métodos heurísticos permitiu que as mesmas instâncias fossem resolvidas de forma quase ótima a um custo computacional mais baixo, aumentando a possibilidade de que a ferramenta possa ser utilizada mais efetivamente na avaliação de múltiplos cenários de utilização.

A metaheurística *Chemical Reaction Optimization* (CRO), proposta recentemente por LAM e LI (2012), foi adaptada para resolver o problema de otimização adotado pela ferramenta. A fim de testar a eficácia do CRO na resolução do problema de otimização adotado, foram utilizadas nos procedimentos de calibração e teste, bases de dados de ampla adoção pela comunidade científica na resolução de problemas de localização semelhantes, como o da p-mediana e p-mediana capacitada. A metaheurística CRO também foi adaptada para resolver tais problemas, de modo que os resultados obtidos pudessem ser comparados com os da literatura.

Finalmente, uma base de dados contendo locais candidatos a receberem estações de carregamento, bem como centros de consumo, foi criada a partir de dados reais de deslocamento veicular obtidos de sistemas de mapeamento, busca e navegação, disponíveis na Internet, para

um centro metropolitano do Brasil. Locais candidatos a receber estações de carregamento foram selecionados de serviços de busca, tomando como base locais em que estações são comumente instaladas em países cuja eletrificação parcial da frota de veículos já é uma realidade. A utilização de distâncias reais de percurso, obtidas de serviços de mapeamento e navegação, também disponíveis na Internet, permitiu uma representação mais precisa das necessidades de consumo de energia dos EVs, em relação ao uso de distâncias euclidianas ou geodésicas.

Espera-se que a presente Dissertação, bem com a ferramenta de suporte à decisão desenvolvida, possam auxiliar futuros esforços de implantação de estações de carregamento de EVs em áreas metropolitanas, contribuindo para o aumento da eletrificação da frota de veículos do Brasil.

1.1 OBJETIVO GERAL

Determinar a localização eficiente de estações de carregamento de veículos elétricos, de forma a minimizar seus custos de implantação, utilizando métodos exatos e heurísticos, bem como deslocamentos reais dos veículos elétricos às estações de carregamento.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver uma ferramenta de suporte à decisão para localização de estações de carregamento de EVs, com o objetivo de minimizar tanto os custos de implantação das estações de carregamento de EVs quanto os custos de deslocamento dos usuários de EVs às estações mais próximas aos mesmos.

- Empregar um modelo de otimização de programação inteira que permita selecionar estações de carregamento de EVs a partir de um conjunto de locais candidatos e um conjunto de centros de consumo, e que considere, além da demanda prevista de carregamento de cada centro, também a energia dispendida por EVs típicos para se deslocarem dos centros de consumo às estações candidatas.

- Empregar um modelo de otimização que possua uma restrição de dispersão de forma a garantir que as estações de carregamento de EVs selecionadas mantenham uma separação mínima entre elas, evitando agrupamentos indesejáveis de estações.

- Empregar um modelo de otimização que restrinja a capacidade máxima de fornecimento de energia das estações de carregamento de EVs, evitando que a demanda a ser suprida pelas mesmas exceda a capacidade de estações comercialmente disponíveis.

- Utilizar rotas reais de deslocamento entre agrupamentos de EVs e estações de carregamento de EVs selecionadas, obtidas de serviços de mapeamento, busca e navegação disponíveis na Internet.

- Adaptar a metaheurística Chemical Reaction Optimization (CRO) (LAM; LI, 2012) para resolução do problema de otimização adotado, a fim de reduzir o custo computacional em relação a resolução por métodos matemáticos exatos, permitindo a avaliação de múltiplos cenários de utilização mais rapidamente.

- Avaliar a eficácia da ferramenta e métodos desenvolvidos na localização de estações de carregamento de EVs em uma região metropolitana.

2 REVISÃO DE LITERATURA

2.1 MODELOS DE LOCALIZAÇÃO DE ESTAÇÕES DE SERVIÇO

O *layout* e planejamento de estações de serviço é um tópico importante e que possui uma variedade de aplicações na vida real. Ambos os setores público e privado da economia são constantemente desafiados com problemas que envolvem decisões de *layout* envolvendo tais estações. Uma parte significativa dos problemas desta natureza envolvem a determinação dos melhores locais para a instalação de estações de serviço, baseado em requerimentos de oferta e demanda. Dentre tais problemas pode-se citar a localização de estabelecimentos comerciais, escolas, hospitais, centros de ambulâncias, estações de bombeiros, caixas eletrônicos bancários, postos de gasolina, estações base de telefonia sem fio, além de estações de carregamento de EVs.

O problema de localização de estações de carregamento de EVs tem atraído muitos esforços de pesquisa nos últimos anos. A maioria dos métodos e modelos de localização para estações de carregamento de EVs são derivados de modelos usados na determinação de locais para instalação de estações de serviço contendo recursos materiais, locais de trabalho e serviços médicos de emergência.

Parâmetros de projeto relativos ao problema de localização de estações são bastante diversos, e podem incluir o número de estações de serviço que devem ser criadas, onde cada estação deve se situar, quão grande cada estação deve ser, além da maneira como a demanda de consumo deve ser alocada à cada estação.

A modelagem do problema de localização de estações tem sido amplamente investigada na literatura. Diferentes tipos de modelos matemáticos para solucionar o problema foram criados, incluindo os modelos *set-covering*, cobertura máxima, p-dispersão, p-centro e p-mediana. Modelos também podem ser categorizados em discretos, planares ou em rede. Modelos estáticos e dinâmicos são também cobertos na literatura. Em modelos estáticos, as entradas de dados para o problema não mudam no tempo, enquanto que nos modelos dinâmicos as entradas são variantes no tempo. Outras categorizações envolvem a demanda, que pode ser elástica ou fixa; as estações que podem ter, ou não, um limite de capacidade e os modelos propriamente ditos, que podem ser determinísticos ou probabilísticos. Finalmente, diferentes métricas foram utilizadas em tais modelos, incluindo a distância Euclidiana (linha reta), distância *Manhattan* e métrica *lp.*

No que diz respeito aos métodos para resolução do problema de localização de estações de serviço, a programação linear e inteira é amplamente utilizada na resolução de tais problemas. Outras abordagens comumente usadas são as providas por metaheurísticas, como a Busca Tabu (ROLLAND et al., 1997), *Simulated Annealing* (OSMAN; CHRISTOFIDES, 1994) e Algoritmo Genético (ALP; ERKUT; DREZNER, 2003). Tais abordagens heurísticas demonstraram ser capazes de obter resultados satisfatórios na solução de problemas particulares de localização, sendo capazes de obter soluções próximas às ótimas em tempo computacional mais baixo que aqueles requeridos por métodos matemáticos exatos.

Segundo Baouche e colaboradores (2014), os modelos de localização de estações se enquadram em três grandes famílias: cobertura (*set-covering*), p-mediana (*p-median*) e p-dispersão (*p-dispersion*).

2.1.1 Modelos de cobertura

Os modelos de cobertura estabelecem um número mínimo de estações para cobrir toda a demanda existente. A população é considerada servida (coberta) quando todas as estações selecionadas estão localizadas dentro de suas distâncias máximas de serviço. O primeiro modelo de cobertura, denominado LSCP (*Location Set Covering Problem*), foi introduzido por Toregas e colaboradores (1971) para localização de ambulâncias. Neste modelo, cada ponto de demanda é coberto por, pelo menos, uma ambulância. Tem como maior limitação o fato de considerar a quantidade de recursos ilimitada, o que raramente acontece na realidade.

Church e Revelle (1974) propuseram uma extensão do problema, denominando-o de *Maximal Covering Location Problem* (MCLP), que tem como objetivo maximizar a quantidade total da população servida, dentro de uma distância máxima de serviço, a partir de um número fixo de estações.

2.1.2 Modelos de p-mediana

Os modelos de p-mediana objetivam minimizar a distância ponderada entre um vértice requisitante e a estação de serviço mais próxima. São amplamente utilizados em redes de computadores, comunicações (localização de antenas) e aplicações militares (centros estratégicos). O modelo mais utilizado foi proposto por Hakimi (1964). Tal modelo prevê que todos os consumidores (vértices requisitantes) sejam atendidos, sendo que cada consumidor é atendido por somente uma estação de serviço.

2.1.2.1 Modelo de p-mediana clássica ou não-capacitada

O problema da p-mediana clássica se constitui em localizar *p* estações de serviço para servir a *n* centros de consumo de forma a minimizar a distância média ponderada entre os centros de consumo e as estações selecionadas mais próximas aos mesmos. As primeiras formulações do problema foram apresentadas por Hakimi (1964). Tal problema é bem conhecido na literatura como sendo NP-difícil (GARY; JOHNSON, 1979). Dessa forma, vários métodos que exploram buscas em árvore foram desenvolvidos de modo a tentar trazer soluções aproximadas para o problema em tempo polinomial. O uso de técnicas de relaxação Lagrangiana e otimização por subgradientes combinadas, de um ponto de vista primal-dual, tem se mostrado como uma boa abordagem para a resolução do problema (GALVÃO; RAGGI, 1989).

Outras abordagens comumente utilizadas são as providas por metaheurísticas, as quais permitem obter rapidamente uma solução aproximada do problema da p-mediana clássica ou, em alguns casos, uma solução ótima, porém sem que seja possível provar a sua otimalidade. Tais heurísticas incluem versões de Busca Tabu (ROLLAND et al., 1997), *Heuristic Concentration* (Rosing e ReVelle, 1997), Algoritmo Genético (ALP; ERKUT; DREZNER, 2003) e *Simulated Annealing* (AL-KHEDHAIRI, 2008).

2.1.2.2 Modelo de p-mediana capacitada

O problema da p-mediana capacitada (CPMP, do inglês *Capacitated p-Median Problem*) se constitui numa variação do problema da p-mediana clássica ou não-capacitada. No CPMP uma demanda fixa é associada a cada centro de consumo e uma capacidade máxima é atribuída a cada estação de serviço candidata. Uma restrição de capacidade é, então, adicionada ao modelo da p-mediana de tal forma que a demanda total dos centros de consumo associados à uma mesma estação não exceda a sua capacidade. O CPMP é considerado NP-difícil (GARY; JOHNSON, 1979).

Mulvey e Beck (1984) foram os primeiros a estender o modelo não-capacitado, ao adicionar restrições de capacidade a cada estação. Em seu trabalho os autores também apresentam uma heurística primal, além de um método hibrido baseado em otimização heurística e subgradientes, a fim de obter boas soluções para o problema formulado. Desde então, diversos outros pesquisadores propuseram novas abordagens que empregam métodos exatos, heurísticos e híbridos para prover soluções de qualidade dentro de tempos computacionais aceitáveis:

- Osman e Christofides (1994) desenvolveram uma solução híbrida baseada na metaheurísticas *Simulated* Annealing e Busca Tabu, a fim de prover soluções ótimas ou quase ótimas para um grupo de instâncias com 50 a 100 vértices e 5 a 10 medianas.

- Maniezzo e colaboradores (1998) apresentaram um método evolutivo combinado com uma técnica de busca local efetiva para solucionar uma variedade de problemas do tipo CPMP, incluindo aqueles propostos por Osman e Christofides (1994).

- Baldacci e colaboradores (2002) propuseram um algoritmo exato para solução do CPMP baseado em particionamento de conjuntos.

- Lorena e Senne (2003) propuseram uma heurística de busca local para ser usada em soluções viáveis obtidas por um processo de otimização Lagrangeana do tipo *surrogate*. Resultados computacionais consideraram instâncias da literatura, bem como dados reais obtidos através de um sistema de informação geográfica.

- Uma versão de Algoritmo Genético foi desenvolvida por Correa e colaboradores (2004).

- Lorena e Senne (2004) desenvolveram uma abordagem de geração de colunas integradas com Relaxação Lagrangeana (do tipo *surrogate*) para cálculo de limites inferiores (*lower bounds*). Resultados computacionais foram apresentados em instâncias criadas a partir de uma base de dados geográfica da cidade de São José dos Campos.

- Ahmadi e Osman (2005) propuseram uma metaheurística denominada *Greedy Random Adaptive Memory Programming Search* (GRAMPS) para o CPMP.

- Um algoritmo do tipo *branch-and-price* para o CPMP foi proposto por Ceselli e Righini (2005).

- Scheuerer e Wendolsky (2006) propuseram um heurística do tipo *scatter search* para o CPMP, sendo que sua eficiência foi analisada em diversas instâncias da literatura, obtendo soluções melhores que as existentes em muitos casos.

- Diaz e Fernandez (2006) propuseram uma metaheurística híbrida baseada em *scatter search* e *path relinking* para o mesmo problema. Os autores conduziram uma série de

experimentos computacionais avaliando o método proposto em instâncias da literatura, incluindo instâncias correspondentes a 737 cidades na Espanha.

- Chaves e colaboradores (2007) apresentaram uma metaheurística híbrida denominada de *clustering search* (CS), que consistia em detectar áreas de busca promissoras baseadas em agrupamentos (*clustering*).

- Boccia e colaboradores (2008) propuseram uma abordagem *cut-and-branch*, a qual provou ser efetiva na resolução ou, pelo menos, na diminuição do *gap* de integralidade de instâncias difíceis utilizando o otimizador CPLEX, da IBM.

- Flezsar e Hindi (2008) desenvolveram uma heurística híbrida, utilizando *Variable Neighborhood Search* (VNS) para achar medianas adequadas e um método exato para resolver o problema de assinalamento resultante, via CPLEX.

- Stefanello e colaboradores (2015) apresentaram uma abordagem "Mateurística", que consistia na redução matemática de modelos por eliminação heurística de variáveis que são improváveis de pertencer a soluções boas ou ótimas. Adicionalmente, um algoritmo de otimização parcial, baseado na técnica de redução apresentada, foi proposto. Os modelos resultantes foram resolvidos pelo CPLEX, com *gaps* pequenos e bom desempenho computacional.

2.1.2.3 Outros modelos de p-mediana

Um modelo considerado por Baouche e colaboradores (2014) como um dos mais realistas, é o FCLP (do inglês, *Fixed Charge Location Problem*) introduzido por Balinski (1965). À cada local candidato, um custo é atribuído para a instalar-se a estação naquele local, assim como uma capacidade máxima de serviço para cada estação. O modelo também introduz um parâmetro que representa o custo de deslocamento de cada consumidor a cada um dos locais candidatos. A função custo consiste de duas partes: a primeira minimiza o número de estações a serem criadas e a segunda, o custo total de deslocamento dos consumidores as estações designadas.

2.1.3 Modelos de p-dispersão

Os modelos de p-dispersão objetivam determinar a localização de servidores, com o objetivo de maximizar sua dispersão. São bastante utilizados na localização de antenas, a fim de maximizar a área coberta e evitar interferências entre elas. Também encontra aplicação na

localização de computadores servidores em redes de comunicação de dados. O modelo enfoca na manutenção de uma distância entre servidores, mas não considera a demanda de consumo. O modelo de Kuby (1987) é um dos mais conhecidos e busca maximizar uma distância mínima entre pares de servidores.

2.1.4 Modelos de localização probabilísticos

Modelos matemáticos de otimização discretos têm mostrado algumas limitações. Isso se deve ao fato de que quando um determinado servidor é atribuído a uma zona de demanda, outras zonas de demanda na mesma área de cobertura podem ficar potencialmente descobertas (DASKIN, 1983). De fato, é mais relevante atribuir vários servidores a uma mesma zona de demanda, a fim de garantir que todas as zonas sejam cobertas, mesmo que um dos servidores esteja ocupado. Para superar esta restrição, várias tentativas se concentraram no problema de atribuição probabilística. Um dos primeiros modelos probabilísticos propostos foi o *Maximum Expected Covering Location*, introduzido por Daskin (1983). O mesmo atribui a cada servidor uma probabilidade *q*, onde *q* é a probabilidade de um servidor estar ocupado.

Várias extensões deste modelo foram criadas, incluindo o TIMEXCLP (REPEDE; BERNARDO, 1994), onde os autores introduzem uma incerteza no tempo de viagem para as ambulâncias, porém, mantendo a estrutura do modelo de Daskin (1983). Revelle e Hogan (1989) propuseram o *Maximum Availability Location Problem* (MALP), como uma versão probabilística do problema *Maximal Covering Location Problem* (MCLP), de Church e Revelle (1974). O MALP posiciona uma quantidade de servidores de tal forma a maximizar a população que irá encontrar um servidor disponível, dentro de um tempo padrão e de uma determinada confiabilidade. O modelo em si é estruturado como um problema de programação inteira binária e resolvido para uma rede de transporte de médio porte em *Baltimore City, Maryland,* EUA.

2.2 MODELOS DE LOCALIZAÇÃO DE ESTAÇÕES DE CARREGAMENTO DE EVS

DAI e colaboradores (2015) patentearam uma invenção que consiste de um método, aparato e sistema para localização de estações de carregamento de EVs, a partir de um conjunto de locais candidatos a receberem tais estações de carregamento e diversos centros de consumo, representados por aglomerados de usuários de EVs dispersos geograficamente. Um módulo de detecção de mobilidade coleta dados de tráfego através de sensores de veículos posicionados em uma determinada área. Um módulo de correspondência cartográfica mapeia o tráfego detectado na mesma área. Um módulo de fluxo de veículos categoriza temporariamente o fluxo

de tráfego mapeado. Um avaliador de requerimentos de EVs determina um número ótimo de estações de carregamento, bem como os respectivos locais para a localização das mesmas. Como resultado da aplicação do método, sistema e aparato, uma relação de estações selecionadas dentre os locais candidatos é fornecida. Também é fornecido um mapa relacionando os centros de consumo com as estações selecionadas. A especificação da invenção permite trabalhar com diversos tipos de sensores de tráfego, além de poder ser implementada em diversos formatos de hardware e software. A invenção utiliza o otimizador CPLEX da IBM (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) para resolução dos modelos de localização. Dependendo do tamanho do problema, expresso em termos do número de estações candidatas e do número de centros de consumo, a resolução via CPLEX pode se constituir numa limitação, devido ao elevado tempo de execução requerido por otimizadores exatos para obtenção de resultados ótimos.

Shao-Yun e colaboradores (2012) desenvolveram um modelo para planejamento de estações de carregamento de EVs em zonas urbanas, que considera a estrutura da malha viária, informações sobre o fluxo de veículos, estrutura do sistema de distribuição de energia, além de restrições de capacidade das estações de carregamento. O modelo proposto seleciona a localização das estações com o objetivo de minimizar o consumo total durante deslocamento dos EVs às mesmas, bem como o investimento nas estações propriamente ditas. O modelo leva em conta tanto os custos para a empresa fornecedora de energia de carregamento quanto para os usuários de EVs. Dessa forma, a função objetivo considera os investimentos na construção das estações, o seu custo de operação e manutenção, o custo das perdas elétricas na rede de distribuição, o custo de carregamento dos EVs nas estações e o custo das perdas energéticas dos EVs no deslocamento às estações de carregamento. A alocação das estações de carregamento é otimizada utilizando a teoria das filas. Os resultados dos estudos de caso mostraram que os métodos e modelo proposto são viáveis para o planejamento de estações de carregamento em áreas urbanas.

Lam, Leung e Chu (2014) estudaram o problema de encontrar os melhores locais para localização de estações de carregamento, do ponto de vista da conveniência para os usuários de EVs e da independência da tecnologia de carregamento. Os requisitos básicos são de que os EVs em circulação por toda a cidade devam ser sempre capazes de alcançar uma estação de carregamento dentro de sua, relativamente baixa, autonomia. O modelo de otimização é independente da tecnologia das estações de carregamento. A conveniência para o usuário é parcialmente modelada através de um fator de desconto α, que expressa a tolerância dos

motoristas em se deslocar até uma estação, para reabastecimento. Além do modelo de otimização propriamente dito, quatro métodos de solução são propostos: Programação inteira linear mista, Programação inteira linear mista efetiva, algoritmo simples do tipo Guloso e metaheurística *Chemical Reaction Optimization* (CRO). O desempenho dos diferentes métodos é comparado em cenários reais e fictícios. O modelo de localização se mostrou eficiente na determinação da localização de estações de carregamento em áreas urbanas de Hong Kong, China.

O problema de cobertura (*set-covering)* tem sido frequentemente adaptado à localização de estações de carregamento de EVs. Merecem destaque as tentativas de Wang (2007, 2008, 2009, 2010) em reformular o problema de cobertura como um problema de programação inteira mista com o objetivo de determinar a localização ideal de estações de carregamento. Frade e colaboradores (2011) propuseram um modelo que maximiza a cobertura de agrupamentos de demanda durante os horários de pico, de manhã e à noite.

Baouche e colaboradores (2014), propuseram uma metodologia para a localização ótima de estações de carregamento de EVs, baseado nos modelos clássicos *Fixed Charge Location* (BALINSKI, 1965) e *p-dispersion* (KUBY, 1987). Os autores se concentraram em minimizar o custo total das viagens entre zonas de demanda e estações de carregamento. A demanda é representada por agrupamentos de EVs, localizados em pontos selecionados. O custo fixo de instalação e/ou operação das estações propriamente ditas também faz parte do modelo, assim como a manutenção de uma distância mínima entre as estações de carregamento. Os autores utilizaram a base de dados de deslocamento veicular VEHLIB (TRIGUI et al., 2004), derivada de pesquisas sobre deslocamentos urbanos feitas na região de *Rhone*, França. A matriz de entrada era composta de 843x843 deslocamentos ponto-a-ponto, sendo depois reduzida a matrizes de deslocamento 42x42, correspondentes a nove distritos de Lyon, França.

O Quadro 1 apresenta as principais contribuições para o domínio da localização de estações de carregamento de EVs, relacionando o tipo de modelo empregado, bem como as suas respectivas restrições.

Quadro 1 – Modelos de localização de estações de carregamento de EVs

(continua)

Autores	Modelo	Objetivos	Restrições de Servidor	Restrições de Demanda	Tipo de servidor	Observação
Wang (2007, 2008)	*Cobertura* (LMLRS)	Minimização do custo de alocação de estações de carregamento	Limite na capacidade dos locais candidatos	Tempo de carga suficiente para completar a viagem; Demanda formulada em termos de viagens a serem satisfeitas	Carregador Tipo I ou II	Motoneta elétrica; Locais de carregamento localizado nos destinos; Utiliza conceito de tempos de carregamento; Carregadores de diferentes tipos numa mesma estação
Wang (2008)	*Cobertura* (BESLM)	Minimização do custo de alocação de estações de carregamento	Restrição na quantidade de veículos sendo carregados e no número de estações	Demanda das viagens de motonetas elétricas devem ser satisfeitas	Troca de bateria	Extensão do LMLRS para sistema de troca de baterias
Wang e Lin (2009)	*Cobertura* (LRVRS)	Minimização do custo de alocação de estações de carregamento	Restrição na quantidade de veículos sendo carregados e no número de estações		Carregamento rápido ou troca de bateria	Extensão do BESLM para veículos de energia alternativa
Wang e Wang (2010)	*Fixed Charge Location* (LPVRS)	Minimização da cobertura máxima e do custo mínimo de alocação	Restrição na quantidade de veículos sendo carregados e no número de estações		Carregamento rápido ou troca de bateria	Melhoria do LRVRS
Frade e colab.. (2011)	*Fixed Charge Location* (LCS)	Maximização da cobertura de agrupamentos de demanda durante o período da manhã e horários de pico	Limite no número de estações de carregamento	Restrições relativas ao número das baias a serem usadas nas estações de carregamento	Carregamento lento no trabalho ou em casa	Modelo de cobertura de demanda. Demanda vinda de modelo que utiliza regressões lineares múltiplas. Estacionamentos em Lisboa, Portugal

Quadro 1 – Modelos de localização de estações de carregamento de EVs

(conclusão)

Autores	**Modelo**	**Objetivos**	**Restrições de Servidor**	**Restrições de Demanda**	**Tipo de servidor**	**Observação**
Xi, Sioshansi, Marano (2013)	*Set Covering* (LPEVCI)	Maximização do número de veículos carregados ou do total de energia de recarga	Restrições de capacidade para os locais escolhidos	Restrições no número de veículos sendo carregados	Estações de carregamento lento	Restrição nos custos de alocação
Chen e colab. (2013)	Localização de estações de carregamento de EVs	Minimização das viagens às estações de carregamento		Satisfação da demanda modelada como áreas de atratividade. Restrições no número de veículos sendo carregados	Estações de carregamento lento	Estações de carregamento lento alocadas em estacionamentos. Estudo do tempo de estacionamento
Andrews e colab. (2012)	Localização de estações de carregamento de EVs	Minimização das viagens de veículos a conjunto de estações de carregamento	Restrições de capacidade nos locais de carregamento. Limite no número de estações que podem ser alocadas	Satisfação da demanda total das viagens dos veículos		Otimização feita sobre um período de *T* intervalos de tempo
Baouche e colab.(2014)	Localização de estações de carregamento de EVs	Minimizar o custo de deslocamento de zonas de demanda as estações de carregamento de EVs, juntamente com os custos de investimento.	Restrições de capacidade nos locais de carregamento. Restrição de dispersão.	Satisfação da demanda total das viagens dos veículos	Carregador Tipo I ou II	Combina modelo *Fixed Charge* com p-dispersão

Fonte: Baouche e colaboradores (2014)

2.3 METAHEURÌSTICA CHEMICAL REACTION OPTIMIZATION (CRO)

Lam e Li (2012) propuseram uma metaheurística para solução de problemas de otimização inspirada na natureza das reações químicas, denominado de *Chemical Reaction Optimization* (CRO). A reação química é um processo natural que transforma substâncias instáveis em estáveis. Numa visão microscópica, uma reação química começa com algumas moléculas instáveis possuidoras de energia excessiva. As moléculas interagem umas com as outras através de uma sequência de reações. Ao final, elas são convertidas naquelas com energia mínima para garantir a sua existência. Esta propriedade é incorporada ao CRO, a fim de resolver problemas de otimização.

A metaheurística CRO se baseia nas duas primeiras leis da termodinâmica (LAM; LI 2012). A primeira lei, a da conservação da energia, diz que a energia não pode ser criada ou destruída, mas apenas transformada de uma forma para outra ou transferida de uma entidade para outra. A segunda lei diz que a entropia, que é a medida do grau de desordem de um sistema, tende a aumentar.

Um sistema de reação química consiste das substâncias químicas e seu ambiente. A energia do ambiente é simbolicamente representada no CRO por um reservatório de energia central (*buffer*). Uma substância química é composta de moléculas, que possuem energia potencial e cinética. Uma reação química ocorre quando o sistema é instável, no sentido de possuir energia excessiva. Todos os sistemas de reação química tendem a alcançar um estado de equilíbrio, no qual a energia potencial cai a um mínimo. O CRO simula esse fenômeno convertendo energia potencial em energia cinética, pela transferência gradual da energia das moléculas para o ambiente, através de passos consecutivos, ou sub-reações, passando por vários estados de transição, que resultam em produtos mais estáveis e com energia mínima. É um processo iterativo em busca do ponto ideal.

Uma alteração química em uma molécula é desencadeada por uma colisão. Existem dois tipos de colisões: unimoleculares e intermoleculares. No CRO, a primeira descreve a situação em que uma molécula colide com a parede de um recipiente, enquanto que a última representa os casos em que uma molécula colide com outras moléculas. Tal alteração química é chamada de reação elementar. Uma reação elementar ineficaz é aquela que resulta em uma mudança sutil da estrutura molecular. O CRO considera quatro tipos de reações elementares: colisão ineficaz com a parede, decomposição, colisão ineficaz intermolecular e síntese (LAM; LI, 2012). Com

respeito às moléculas, a decomposição e a síntese estão relacionadas com a colisão ineficaz com a parede e a colisão ineficaz intermolecular, respectivamente, mas provocam mudanças muito mais vigorosas nas estruturas moleculares. As reações moleculares elementares estão resumidas no Quadro 2 e ilustradas na Figura 1.

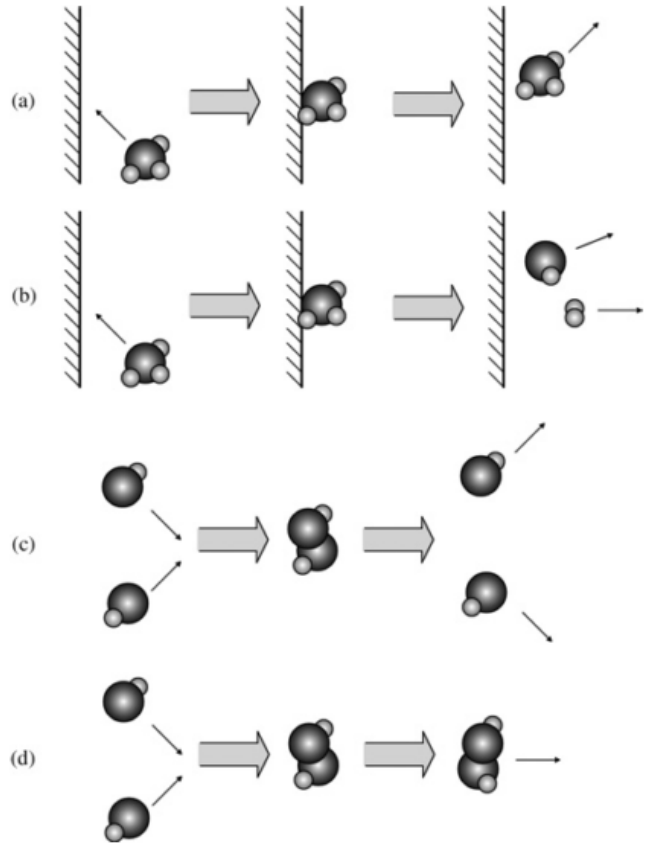
Soluções são manipuladas através de uma sequência aleatória de reações. As duas colisões ineficazes implementam busca local (intensificação), enquanto que a decomposição e a síntese dão o efeito de diversificação. Uma mistura adequada de intensificação e diversificação permite fazer uma busca efetiva do mínimo global dentro do espaço de solução.

Quadro 2 – Características das quatro reações elementares

Características das quatro reações elementares		
	Número de moléculas envolvidas	
Extensão da mudança	**Unimolecular**	**Intermolecular**
Maior	Decomposição	Síntese
Menor	Colisão ineficaz com a parede	Colisão ineficaz intermolecular

Fonte: Lam e Li (2012)

Figura 1 – Quatro reações elementares do CRO



(a) Colisão ineficaz com a parede; (b) Decomposição;
(c) Colisão ineficaz intermolecular; (d) Síntese.
Fonte: Lam e Li (2012)

O CRO é uma metaheurística populacional variável. Desta forma, o número total de moléculas em diferentes iterações pode não ser o mesmo. Nas colisões ineficazes, o número de moléculas antes e depois da alteração permanece idêntico. Já na decomposição esse número aumenta e na síntese, diminui. É possível influenciar indiretamente as frequências de decomposição e síntese alterando os parâmetros do CRO denominados α e β, respectivamente. As reações elementares definem as implementações (algoritmos) das interações moleculares.

O CRO foi aplicado, com sucesso, para resolver os problemas em ambos os domínios discreto e contínuo, incluindo o problema da atribuição quadrática (XU et al., 2010), treinamento em rede neural (JAMES et al., 2011), problemas contínuos multimodais (LAM et al., 2012), dentre outros. No entanto, até a presente data, não é do conhecimento do autor que o CRO tenha sido utilizado na resolução do problema de localização de estações de carregamento de EVs, utilizando o modelo proposto por Baouche e colaboradores (2014).

2.4 VEÍCULOS ELÉTRICOS

Um veículo elétrico ou EV, do inglês *Electric Vehicle*, é aquele que emprega um ou mais motores elétricos para sua propulsão. Pode ser alimentado por um sistema que obtém eletricidade de fontes fora do veículo ou pode ser autônomo com uma bateria, painéis solares ou um gerador elétrico para converter combustível em eletricidade (FAIZ et al., 1996). Os EVs incluem veículos rodoviários e ferroviários, embarcações de superfície, submarinas e aviões elétricos.

Os EVs de deslocamento terrestre, mais especificamente os automóveis, podem ser categorizados em veículos elétricos híbridos (HEVs, do inglês *Hybrid Electric Vehicle*) e veículos elétricos plug-in (PEVs, do inglês *plug-in Electric Vehicles*).

2.4.1 Veículos elétricos híbridos (HEVs)

Um veículo elétrico híbrido é aquele que combina um trem de força convencional com alguma forma de propulsão elétrica. O trem de força geralmente é composto de um motor a combustão interna (ICE, do inglês *Internal Combustion Engine*) com potência inferior à de um automóvel convencional, aliado a um pequeno motor elétrico, cuja finalidade é a de alcançar uma maior economia de combustível ou, em alguns modelos, atingir um melhor desempenho. Também há uma significativa redução nas emissões quando comparado à um automóvel movido exclusivamente a gasolina ou álcool, notadamente em zonas urbanas, onde o motor elétrico trabalha quase que exclusivamente.

Na maioria dos HEVs, o ICE está conectado diretamente às rodas do automóvel e também à um gerador elétrico que permite carregar suas baterias. As baterias podem também ser carregadas por frenagem regenerativa, um processo no qual a energia cinética do veículo durante frenagens é convertida em energia elétrica. A capacidade da bateria nos HEVs é bem mais modesta que nos PEVs, girando em torno dos 5 kWh, contra 20 ou mais dos PEVs. Segundo testes do Departamento de Energia dos Estados Unidos, a bateria do HEV *Toyota Prius*, modelo 2013, é capaz de fornecer 4.1kWh (US ENERGY DEPARTAMENT, 2013).

Os HEVs são, de longe, os automóveis elétricos mais vendidos no mundo, tendo sido comercializados mais de 12 milhões de veículos, desde 1997 (COBB, 2016).

2.4.2 Veículos elétricos plug-in (PEVs)

Um veículo elétrico plug-in (PEV) é aquele veículo a motor que pode ser recarregado de qualquer fonte externa de eletricidade, desde simples tomadas residenciais a carregadores dedicados de corrente contínua. A energia elétrica armazenada em conjuntos de bateria recarregáveis fornece a energia total ou parcial, necessária para acionar as rodas motrizes. PEVs são comumente categorizados em veículos totalmente elétricos ou a bateria (BEVs) e veículos híbridos plug-in (PHEVs) (SUTOR; HUDGINS, 2016).

2.4.2.1 Veículos elétricos híbridos plug-in (PHEVs)

Os PHEVs combinam dois modos de propulsão em um só veículo: um motor elétrico que é alimentado por baterias, sendo que as mesmas podem ser carregadas diretamente da rede elétrica, além de um ICE que pode ser reabastecido com gasolina (DRIVE CLEAN, 2016). O ICE é acionado somente quando absolutamente necessário, ou seja, quando as baterias estão praticamente sem carga, durante acelerações rápidas ou quando ao ar-condicionado ou aquecimento são usados mais intensamente.

Alimentar o veículo com eletricidade vinda da rede de distribuição elétrica reduz custos operacionais, consumo de petróleo, além de reduzir emissões quando comparado com veículos convencionais do tipo ICE. Quando as distâncias a serem percorridas excedem o limite em que o PHEV consegue rodar somente com motores elétricos, o mesmo se comporta como um HEV, ou seja, um motor ICE gerará energia para carregamento das baterias e acionamento dos motores elétricos ou acionará diretamente as rodas motrizes.

As baterias dos PHEVs são, em geral, maiores que a dos HEVs, o que proporciona aos mesmos uma autonomia, rodando somente com motores elétricos, de cerca de 10 a 80 milhas. Tal autonomia permite que, num cenário típico, um condutor de PHEV possa se deslocar de casa até o trabalho, opcionalmente carregar as baterias parcialmente durante o horário de trabalho e retornar a casa, onde poderá fazer o carregamento noturno das baterias (SUTOR; HUDGINS, 2016) . Neste cenário o motor ICE não precisaria ser acionado em momento algum, permitindo uma significativa economia de combustível com emissões próximas a zero.

Existem duas categorias de arquiteturas utilizadas por PHEV, que diferem na maneira que combinam a energia do motor elétrico com a do ICE:

- Paralela: nesta arquitetura tanto o motor elétrico quanto o ICE estão acoplados mecanicamente as rodas motrizes. Num dado momento, tanto o motor elétrico quanto o ICE podem tracionar as rodas diretamente.

- Série: nesta arquitetura, apenas o motor elétrico está conectado mecanicamente as rodas motrizes. O ICE é usado apenas para gerar eletricidade. O PHEV *Chevy Bolt*, mostrado na Figura 2, emprega uma versão ligeiramente modificada desta arquitetura, onde o motor elétrico traciona as rodas a maior parte do tempo, mas, o ICE pode ocasionalmente também tracionar diretamente as rodas, como na arquitetura paralela, quando o veículo estiver desenvolvendo altas velocidades em autoestradas ou quando as baterias estiverem completamente descarregadas (SUTOR; HUDGINS, 2016).

Figura 2 – Automóvel PHEV *Chevy Bolt*, modelo 2018

Fonte: General Motors Corp. (2018)

2.4.2.2 Veículos elétricos a bateria (BEVs)

Os veículos a bateria (BEVs, do inglês *Battery Electric Vehicles*) são aqueles que utilizam somente baterias para alimentar um ou mais motores elétricos. Também são denominados simplesmente de EVs, por serem totalmente elétricos, ou seja, não contarem com ICEs para geração de energia elétrica ou tracionamento das rodas motrizes. A fim de preservarem ao máximo a carga de suas baterias, todos os BEVs contam com frenagem regenerativa

De acordo com a Administração Federal de Rodovias (FHWA), do Departamento de Transportes dos EUA, a autonomia típica de um BEV, com baterias totalmente carregadas, é de cerca de 100 milhas, porém, em alguns BEVs, a mesma pode chegar a 265 milhas (TURCHETTA, 2017). Também de acordo com a FHWA, essa autonomia é suficiente para atender a mais de 90% de todos os deslocamentos diários nos EUA. Em deslocamentos mais longos, os BEVs deverão ser recarregados, o que pode levar de 15 minutos até quase um dia inteiro dependendo do estado de carga das baterias, do seu tipo e tamanho e, principalmente, do equipamento utilizado para recarga. A Tabela 1 mostra a autonomia, o preço sugerido de aquisição e o custo para se obter 1 milha de autonomia, para diversos modelos de BEVs disponíveis nos EUA, em 2017. Os valores estão em dólares americanos ($USD).

Tabela 1 – Autonomia de BEVs, por modelo vendido nos EUA em 2017

Modelo de BEV	**Autonomia (Milhas)**	**Preço ao consumidor ($USD)**	**Custo por milha de autonomia ($USD/Milha)**
2017 Mitsubishi i-MiEV	63	$23.500,00	$373,02
2017 Smart Fortwo electric drive	70	$25.750,00	$367,86
2017 Ford Focus Electric	76	$29.120,00	$383,16
2017 FIAT 500e	84	$32.995,00	$392,80
2017 Kia Soul EV	93	$32.250,00	$346,77
2017 Nissan Leaf	107	$31.000,00	$289,72
2017 BMW i3	114	$42.400,00	$371,93
Tesla S 70	234	$72.700,00	$310,68
2017 Chevrolet Bolt	238	$36.620,00	$153,87
Tesla X 75D	238	$85.500,00	$359,24
Tesla X 90D	257	$93.500,00	$363,81
Tesla X P100D	289	$145.000,00	$501,73
Tesla S 100D	360	$97.500,00	$270,83

Fonte: Schmidt (2017)

Na presente dissertação, o termo EV será usado para designar os veículos elétricos do tipo BEV, já que este é o objeto de estudo da mesma, por sempre necessitar de uma fonte externa de energia elétrica para recarga de suas baterias.

2.5 BATERIAS DE EVS

2.5.1 Tecnologia das baterias de EVs

A maioria dos EVs atualmente no mercado usa baterias de íon-lítio (*Li-Ion*), amplamente conhecidas por seu uso em *laptops*, *smartphones* e diversos outros eletrônicos de consumo. No entanto, diversos esforços estão em andamento para desenvolver e testar novas químicas de bateria, que poderão ajudar a reduzir as preocupações dos consumidores quanto a ainda restrita autonomia e elevado preço final do veículo (TURCHETTA, 2017).

No passado, já foram utilizadas, em HEVs, baterias de bateria de hidreto metálico de níquel (NiMH). Porém, a baixa energia específica em relação as baterias de *Li-Ion* (quase a metade) e a presença de efeito memória, acabaram por determinar o fim do seu uso.

A capacidade de armazenamento de energia de um EV moderno típico varia entre 24 e 36 kWh (DRIVE CLEAN, 2016).

2.5.2 Estações de carregamento de EVs

Uma estação de carregamento de veículos elétricos, também denominada EVSE (do inglês, *Electric Vehicle Supply Equipment*), é um elemento de infraestrutura que fornece energia elétrica para recarregar PHEVs e BEVs, transferindo energia de forma segura entre a rede de distribuição elétrica e os mesmos (ELECTRIC TRANSPORTATION ENGINEERING CORPORATION, 2010).

O fornecimento de energia pelo EVSE pode ser feito em corrente alternada (CA) ou corrente contínua (CC). No primeiro caso, um carregador, localizado no interior do veículo, efetua a conversão de CA para CC, além de controlar o carregamento das baterias. Quando o carregamento é feito por CC, o carregador é desativado, sendo o controle de carregamento feito pelo próprio EVSE.

2.5.2.1 Tipos de carregamento

Existem atualmente 4 tipos de cenários, ou níveis de serviço, de carregamento de EVs, conforme definidos pela *Society of Automotive Engineers* (SAE) (Saxton, 2011):

- Nível 1: Este nível de serviço é fornecido por uma tomada residencial típica de 120VCA, e é mais apropriado para PEVs com conjuntos de bateria relativamente pequenos, de baixa

quilometragem diária ou acesso limitado ao carregamento de Nível 2. A Figura 3 mostra uma estação de carregamento residencial de nível 1 típica, que consiste basicamente de um cabo com conector padrão J1772 (utilizado nos EUA) e um circuito de proteção (ALLSTATE ELECTRIC, 2017).

Figura 3 – Estação de carregamento nível 1

Fonte: AllState Electric (2017)

- Nível 2: Este nível de serviço é apropriado para carregar completamente a maioria dos BEVs e PHEVs durante a noite, e opera com tensões nominais entre 220 e 240VCA em circuito monofásico. Em comparação com o nível 1, ele pode reduzir à metade o tempo de carregamento, que gira em torno de 4 a 6 horas para uma recarga completa. O carregamento de nível 2 pode exigir que os proprietários de EVs atualizem seu quadro elétrico, a fim de fornecer um circuito dedicado para o carregamento do EV. Este tipo de carregamento também é oferecido em mais de 14.000 locais públicos e privados nos EUA, como estacionamentos, shopping centers, aeroportos, e outros (LAMBERT, 2017). O mesmo conector, do tipo J1772, mostrado na Figura 4, é usado para os carregamentos de níveis 1 e 2, sendo que a maioria dos EVs modernos são compatíveis com ambos os níveis de tensão (120 – 240VCA). Uma estação de carregamento de nível 2, localizada em um estacionamento de um aeroporto, é mostrada na Figura 5.

Figura 4 – Conector SAE J1772 com respectiva tomada veicular de carregamento

Conector SAE J1772 Tomada de carregamento J1772

Fonte: Electric Transportation Engineering Corporation (2010)

- Nível 3: Também conhecido como carregamento rápido em corrente contínua, este nível de carregamento pode satisfazer as necessidades de carregamento de locais comerciais de alto volume de tráfego, como postos de gasolina, concentração de frotas de veículos, além de estações de carregamento ao longo dos principais corredores de transporte.

  Uma estação de nível 3 típica pode fornecer cerca de 64 Km de autonomia a um EV com apenas 10 minutos de carregamento. Com 15 a 30 minutos a bateria poderá ser carregada completamente, proporcionando de 96 a 128Km de autonomia (SAXTON, 2011). Os carregadores nível 3 operam com tensões de saída de 208 a 600 VCC. A Tabela 2 sumariza os diversos níveis de carregamento disponíveis, seu tipo de uso, bem como corrente e tensão de saída dos carregadores.

- Troca de bateria: um processo automatizado que permite a troca de uma bateria esgotada por uma totalmente carregada.

Figura 5 – Estação de carregamento nível 2

Fonte: Turchetta (2017)

Figura 6 – Estação de carregamento nível 3

Fonte: Saxton (2011)

Tabela 2 – Opções de carregamento, por nível

Nível	**CC /CA**	**Corrente de saída (A)**	**Tensão de Saída (V)**	**Potência (kW)**	**Autonomia (Km / 30 minutos de carga)**	**Uso típico**
Nível 1	CA	12 a 16	120	1.3 a 1.9	1.6 a 4	Residencial
Nível 2	CA	Até 80	240	Até 19.2	8 a 16	Residencial e público
Nível 3	CC	Até 200	208 a 600	50 a 150	96 a 128	Público

Fonte: Turchetta (2017)

Como resultado da revisão de literatura efetuada, verificou-se que nem o estado da arte e nem o estado da técnica descrevem nenhum método, aparato ou sistema específico para localização de estações de carregamento de EVs, utilizando a metaheurística CRO e que leve em conta a energia dispendida no deslocamento dos EVs às estações, seus custos de instalação e manutenção e a distância mínima entre estações de carregamento.

3 METODOLOGIA

A metodologia empregada nesta dissertação consiste de doze etapas, delineadas no Quadro 3, a saber: a definição de cenários de carregamento de EVs; a seleção de um modelo de otimização e bases de dados de teste e validação; a criação de bases de dados com deslocamentos veiculares reais; a adaptação da metaheurística CRO para resolução do problema de localização, além da condução de uma série de testes e análises, com o objetivo de avaliar a qualidade das soluções encontradas.

Os itens a seguir descrevem, em detalhes, cada etapa da metodologia adotada:

Quadro 3 – Etapas da Metodologia

Etapa	Título	Seção
1	Definir os cenários de carregamento de EVs, em um centro metropolitano.	3.1
2	Escolher um modelo de otimização para a localização de estações de carregamento de EVs que atenda os cenários definidos na etapa 1.	3.2
3	Selecionar bases de dados científicas de referência, contendo problemas semelhantes ao escolhido na etapa 2, a fim de testar a eficácia da adaptação do CRO.	3.3
4	Importar as bases de dados da etapa 3 para uso pela metaheurística CRO e por um otimizador matemático comercial.	3.4
5	Codificar o problema de otimização da etapa 2, na linguagem de programação do otimizador matemático comercial.	3.5
6	Adaptar a metaheurística CRO para a resolução do problema da etapa 2, além dos problemas das bases de dados selecionadas na etapa	3.6
7	Resolver problemas das bases de dados selecionadas na etapa 3, de forma ótima, por um otimizador comercial, a fim de obter uma linha de base de desempenho (*gaps* e tempos de execução) para avaliação da metaheurística CRO.	4
8	Resolver os problemas das bases de dados da etapa 3 utilizando CRO.	4
9	Analisar os resultados obtidos pelo CRO, na etapa 8, comparando-os com os resultados obtidos na etapa 7 e os resultados da literatura.	4
10	Criar uma base de dados, composta de agrupamentos de EVs e estações de carregamento candidatas, a partir de dados obtidos de serviços de mapeamento, busca e navegação, disponíveis na Internet.	3.7
11	Resolver os problemas das bases de dados da etapa 10 para os cenários definidos na etapa 1, utilizando o modelo da etapa 2, pela metaheurística CRO e por um otimizador matemático comercial.	4
12	Analisar os resultados obtidos na etapa 11.	4

Fonte: Elaborado pelo autor (2017)

3.1 DEFINIR OS CENÁRIOS DE CARREGAMENTO DE EVS, EM UM CENTRO METROPOLITANO

Para avaliar o desempenho da ferramenta de suporte à decisão, uma base de dados contendo locais candidatos a receberem estações de carregamento de veículos elétricos em um centro metropolitano foi criada, utilizando o serviço online de mapeamento *Google Maps API* (GOOGLE INC., 2018). O centro metropolitano escolhido foi a Grande Vitória no estado do Espírito Santo, Brasil. A Grande Vitória compreende as cidades de Vitória, Viana, Serra, Cariacica, Vila Velha, Fundão e Guarapari e possuía, em 2016, mais de 870 mil carros e 400 mil motocicletas (CONFEDERAÇÃO NACIONAL DO TRANSPORTE, 2016).

Inicialmente, foram selecionados 272 pontos de interesse, que poderiam ser candidatos a instalação de estações de carregamento EVs. Através da *Google Maps API*, foram selecionados postos de gasolina, hospitais, estacionamentos comerciais, shopping centers, além do aeroporto de Vitória. Tais pontos de interesse correspondem a locais típicos onde estações comerciais de carregamento de EVs têm sido instaladas nos Estados Unidos (MCCAULEY, 2017).

Devido a eletrificação da frota brasileira de automóveis ser praticamente inexistente, informações sobre o perfil dos consumidores de EVs no Brasil, bem como dos custos para a aquisição e instalação de estações de carregamento de EVs não estavam disponíveis quando da escrita da presente Dissertação. Desta forma, para que os testes e validação da ferramenta de suporte à decisão pudessem ser levados a cabo, utilizaram-se alguns valores típicos, baseados em informações disponíveis nos EUA e países da Europa.

O cenário utilizado nos testes e validação está descrito a seguir:

- Frota: 100 a 1000 EVs, correspondendo de 0,01% a 0,1% da frota atual estimada de veículos da Grande Vitória (1 milhão). Nos testes para avaliação da eficácia da versão adaptada do CRO, quando comparado a métodos exatos, utilizou-se de 200 a 1000 EVs, dependendo da instância.

- BEVs: Conjunto de baterias com capacidade de 24 kWh, correspondente a menor capacidade de bateria dentre os BEVs a venda nos EUA, em 2018 (EVADOPTION, 2018). Consumo de energia combinado (cidade e estrada) de 0,01865 kWh/Km, correspondente a média dos 14 principais BEVs a venda nos EUA, também em 2018.

- Estações de carregamento de EVs: Foram escolhidas estações com carregadores de nível 3 (carga rápida em CC), por serem os únicos que permitem carregamento completo das baterias da maioria dos EVs em menos de 30 minutos. Além disso, as estações de carregamento de nível 3 são as que apresentam maior taxa de crescimento atualmente, tendo gradualmente substituído as de nível 2, nos EUA (LAMBERT, 2017). Cada local candidato conterá quatro carregadores duais, ou seja, com 2 conectores, permitindo o carregamento de dois EVs simultaneamente. A capacidade de fornecimento é de 100kWh (50kWh por conector), com disponibilidade de 12h diárias, perfazendo uma capacidade total diária de 4.800 kWh, por estação. Custo de 150.000 Reais, por carregador, ou aproximadamente, 50.000 dólares americanos (AGENBROAD; HOLLAND, 2014). Portanto, o custo total, por estação, é de 600.000 Reais.

Maiores detalhes relativos aos cenários desenvolvidos para a presente dissertação podem ser encontrados na Etapa 3.7.

3.2 ESCOLHER UM MODELO DE OTIMIZAÇÃO PARA A LOCALIZAÇÃO DE ESTAÇÕES DE CARREGAMENTO DE EVS QUE ATENDA OS CENÁRIOS DEFINIDOS NA ETAPA 3.1.

O modelo escolhido para localização de estações de carregamento de EVs foi aquele desenvolvido por Baouche e colaboradores (2014). A razão da escolha se deve ao fato do mesmo retratar com bastante fidelidade as necessidades tanto do ponto de vista dos usuários de EVs, quanto dos futuros proprietários de estações de carregamento. Outras características desejáveis do modelo são:

- O modelo utiliza uma representação que permite que o mesmo seja resolvido, mesmo que não estejam inteiramente disponíveis todas as informações utilizadas por Baouche e colaboradores (2014), o que o torna bastante indicado para o presente estado de eletrificação da frota brasileira, que é quase nulo. Por exemplo, uma das variáveis do modelo determina o número de EVs se deslocando para um centro de consumo. Na ausência de tal informação pode-se assumir um valor fixo, de tal forma que os testes de validação previstos possam ser conduzidos.

- Com a adição de apenas uma restrição e o ajuste de valores de dois dados de entrada do modelo, correspondentes à distância mínima de separação entre estações de carregamento e a capacidade de fornecimento de energia de uma dada estação, o modelo passa a se

comportar como o modelo da p-mediana ou p-mediana capacitada. Isto permite que bases de dados padrão da literatura possam ser utilizadas nos testes de validação da codificação do modelo no CPLEX e da adaptação do CRO.

O modelo de Baouche e colaboradores (2014) é baseado nos modelos clássicos FCLP - *Fixed Charge Location* (BALINSKI, 1965) e *p-dispersion* (KUBY, 1987). Os autores se concentraram em minimizar o custo total das viagens entre zonas de demanda e estações de carregamento. A demanda é representada por agrupamentos de EVs, localizados em pontos selecionados. O custo fixo de instalação das estações propriamente ditas também faz parte do modelo, assim como a manutenção de uma distância mínima entre as estações de carregamento.

O modelo Baouche e colaboradores (2014) é mostrado a seguir:

Variáveis de decisão:

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

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

Função de otimização:

$$Min \sum_{j \in J} f_j\ x_j + \propto \sum_{i \in I} \sum_{j \in J} n_i^{(ev)} d_{ij}\, y_{ij} \quad (3)$$

Sujeito a:

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

$$y_{ij} - x_j \leq 0, \forall\ i\ \in I, \forall\ j\ \in J \quad (5)$$

$$\sum_{i \in I} n_i^{(ev)} (Di + d_{ij}) y_{ij} \leq C_j x_j, \forall\ j\ \in J \quad (6)$$

$$r x_i x_j \leq dist_{ij}, \forall\ i, j\ \in I \quad (7)$$

$$x_j\ \in \{0,1\}, \forall\ j\ \in J, r > 0 \quad (8)$$

$$y_{ij}\ \in \{0,1\},\ \forall\ i\ \in I, \forall\ j\ \in J \quad (9)$$

Onde:

J : conjunto de locais candidatos a receber estações de carregamento de EVs;

I : conjunto de centros de consumo;

f_j : custo para se situar uma estação de carregamento no local candidato *j;*

D_i : demanda de energia no centro de consumo *i*;

d_{ij} : Energia necessária para um veículo se deslocar de um centro de consumo *i* a uma estação de carregamento *j;*

r : distância mínima entre duas estações de carregamento;

$dist_{ij}$: distância entre a estação *i* e a estação *j;*

C_j : capacidade da estação de carregamento *j*;

$\propto$: custo do quilowatt-hora, que pode ser multiplicado por um período de tempo *T*, necessário para obtenção de lucro;

$n_i^{(ev)}$: número médio de veículos se deslocando para um centro de consumo *i;*

A função (3) representa a função de otimização. O objetivo é minimizar o custo total de instalação e manutenção, bem como a energia total dispendida por todos os veículos para chegar as estações de carregamento designadas. As restrições (4) e (5) asseguram que cada agrupamento de demanda seja coberto por uma estação de carregamento. A restrição (6) garante que a demanda atribuída a estação *j* não esteja além da capacidade C_j, daquela estação de carregamento. A restrição (7) é derivada do problema de p-dispersão (KUBY, 1987). Esta restrição não-linear força cada estação a estar separada por uma distância mínima *r*. A mesma pode ser simplificada para uma restrição linear, como mostra a equação (10), abaixo:

$$r + (M - dist_{ij})\, x_i + (M - d_{istij})x_j \leq 2M - dist_{ij} \qquad (10)$$

Onde *M* é uma constante de valor alto, como *Max* (*dist*). Finalmente, as restrições (8) e (9) definem y_{ij} e x_j como binários (0 ou 1) e *r* como sendo maior que 0.

As variáveis de decisão, bem como a função objetivo, representam as saídas do modelo de programação inteira. As variáveis de decisão *x* contém as estações selecionadas. Por exemplo, se x_0 contivesse o valor um, significaria que o primeiro local candidato a se tornar uma estação de carregamento foi selecionado. As variáveis de decisão *y* contém o mapeamento entre o centro de consumo e as estações de carregamento selecionadas. Por exemplo, se y_{34} contivesse o valor um, significaria que o centro de consumo três foi associado à estação de carregamento quatro, e assim sucessivamente.

Uma vez que Baouche e colaboradores (2014) não atribuíram um nome ou sigla especifica a seu modelo de otimização, na presente dissertação o mesmo será denominado FCLP Baouche.

3.3 SELECIONAR BASES DE DADOS CIENTÍFICAS DE REFERÊNCIA, CONTENDO PROBLEMAS SEMELHANTES AO ESCOLHIDO NA ETAPA 3.2, A FIM DE TESTAR A EFICÁCIA DA ADAPTAÇÃO DO CRO

Nesta etapa, bases de dados de referência foram selecionadas da literatura, a fim de testar a eficácia da adaptação do CRO na resolução do modelo de localização escolhido. Não foi possível utilizar a base de dados empregada por Baouche e colaboradores (2014), uma vez que a mesma não foi disponibilizada em seu artigo. Da mesma forma, não foi encontrada na literatura nenhuma outra base de dados que tivesse sido especificamente criada para a resolução pelo modelo proposto por Baouche e colaboradores (2014).

Com o objetivo de resolver o problema da indisponibilidade de bases específicas para o problema escolhido, selecionaram-se bases de dados que tivessem sido criadas para serem resolvidas por modelos matemáticos similares ao escolhido. Uma condição para a seleção de tais bases foi a de que nenhuma restrição tivesse que ser removida do modelo escolhido, a fim de não o descaracterizar. Outra condição foi de que não mais do que uma restrição tivesse que ser adicionada, a fim de adequar o modelo escolhido às bases de dados de referência selecionadas. Por fim, as bases deveriam ser recentes ou amplamente referenciadas na literatura, já tendo sido resolvidas por diversas heurísticas e métodos exatos, para que o desempenho da adaptação da metaheurística CRO pudesse ser avaliada por comparação com outros métodos.

Após revisão de modelos e bases compatíveis com os requerimentos supracitados, quatro bases de dados, contendo um total de 81 problemas (instâncias), foram selecionadas da literatura. A primeira das bases contém problemas de p-mediana não-capacitada (HAKIMI,1964) e as outras cinco, problemas de p-mediana capacitada (MULVEY; BECK, 1984).

A seguir estão descritas as modificações efetuadas no modelo de Baouche e colaboradores (2014) para compatibilizá-lo com os modelos e bases selecionadas da literatura:

3.3.1 Modelo de p-mediana não-capacitada

3.3.1.1 Adequações promovidas no modelo FCLP Baouche

A seguir estão relacionadas as principais adequações feitas no modelo original de Baouche e colaboradores (2014) para que o mesmo pudesse se comportar como o modelo de p-mediana não-capacitada:

- Restrição do número de estações abertas (p) : No modelo da p-mediana, o número de estações de carregamento é fixo e igual a p. Desta forma uma nova restrição foi adicionada ao modelo de Baouche e colaboradores (2014):

$$\sum_{j=1}^{n} x_j = p \,, \forall\, j \in J \tag{11}$$

- Restrição de capacidade (6): No modelo da p-mediana não-capacitada não existem restrições quanto à capacidade de fornecimento das estações de serviço, ou seja, considera-se que o limite de fornecimento de energia das mesmas é ilimitado. Para que o modelo de Baouche e colaboradores (2014) possa se comportar como o da p-mediana, deve-se tornar a restrição (6) sem efeito. Para tanto, foram atribuídas às constantes C_j, $\forall\, j \in J$, que representam as capacidades das estações de carregamento, um valor muito alto correspondente ao "*infinito"* na linguagem de programação utilizada na implementação do CRO.

- Restrição de p-dispersão (7) (10): o modelo da p-mediana não conta com a restrição de p-dispersão, presente no modelo de Baouche e colaboradores (2014), devendo a mesma ser tornada sem efeito. Para isso à constante r, que representa a distância mínima entre duas estações de carregamento, foi atribuído o valor zero.

- f_j : Ao custo para se situar uma estação de carregamento no local candidato j, $\forall\, j \in J$, foi atribuído o valor zero, tornando o primeiro termo ($\sum_{j \in J} f_j\ x_j$) da função objetivo (3) sem efeito, já que o mesmo não faz parte do modelo da p-mediana não-capacitada.

- Di: À demanda de energia no centro de consumo i, $\forall i \in I$, foi atribuído o valor zero, já que (6) não faz parte do modelo da p-mediana não capacitada.

- $\propto$: Ao custo do quilowatt-hora, foi atribuído o valor 1, uma vez que não faz parte do modelo da p-mediana não capacitada.

- $n_i^{(ev)}$: Ao número médio de veículos se deslocando para um centro de consumo i, $\forall i \in I$, foi atribuído o valor 1, uma vez que o mesmo não faz parte do modelo da p-mediana não capacitada.

3.3.1.2 Modelo de localização modificado

O modelo da p-mediana não-capacitada, resultante das adequações feitas ao modelo de Baouche e colaboradores (2014), incluindo a remoção de todas as inequações e termos que foram tornados sem efeito, é mostrado abaixo:

Variáveis de decisão:

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

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

Função de otimização:

$$Min \sum_{i \in I} \sum_{j \in J} d_{ij}\, y_{ij} \quad (12)$$

Sujeito a:

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

$$y_{ij} - x_j \leq 0, \forall\, i \in I, \forall\, j \in J \quad (5)$$

$$x_j \in \{0,1\}, \forall\, j \in J, r > 0 \quad (8)$$

$$y_{ij} \in \{0,1\}, \quad \forall\, i \in I, \forall\, j \in J \quad (9)$$

$$\sum_{j=1}^{n} x_j = p\ , \forall\, j \in J \quad (11)$$

Onde:

J : conjunto de locais candidatos a receber estações de serviço;

I : conjunto de centros de consumo;

d_{ij} : energia necessária para um veículo se deslocar de um centro de consumo *i* a uma estação de carregamento *j;*

p : número de estações a serem abertas (alocadas);

A função (12) representa a função de otimização. O objetivo é minimizar o custo total ponderado pela energia de deslocamento. A restrição (11) estabelece que o número de estações

a serem abertas seja igual a *p*. A restrição(4) estabelece que toda a demanda do centro de demanda i deve ser satisfeita. A restrição (5) assegura que cada centro de demanda só possa ser associado a uma estação que tenha sido selecionada. Finalmente, as restrições (8) e (9) estabelecem que as variáveis de decisão devem ser binárias (0 ou 1) e não-negativas.

3.3.1.3 Base de dados e experimentos realizados

A base de dados de p-mediana não-capacitada selecionada foi disponibilizada por Beasley (1985) em sua biblioteca online de problemas de pesquisa operacional, denominada *OR-Library* (BEASLEY, 1990) e consiste de 40 problemas, nomeados de pmed1 a pmed40, com tamanhos de 100 a 900 estações servidoras candidatas / consumidores e com um número fixo de estações servidoras selecionadas variando de 5 a 100. Trata-se de uma base clássica, que tem sido amplamente utilizada na literatura por mais de três décadas.

Os resultados obtidos pela presente implementação do CRO foram comparados, em tempos de *gaps* e tempo de execução, com aqueles obtidos pelo otimizador CPLEX da IBM (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) e também com aqueles obtidos pelas seguintes metaheurísticas:

- Relaxação Lagrangiana (RL): resolução dos 40 problemas da *OR-Library* por Daskin e Maass (2015).
- *Variable Neighborhood Search* (VNS): resolução de 22 problemas da *OR-Library* por Hansen e Mladenović (1997).
- Algoritmo Genético (GA): resolução dos 40 problemas da *OR-Library* por Alp, Erkut e Drezner (2003).

3.3.2 Modelo da p-mediana capacitada

3.3.2.1 Adequações promovidas no modelo FCLP Baouche

A seguir estão relacionadas as principais adequações feitas no modelo de Baouche e colaboradores (2014) para que o mesmo pudesse se comportar como o modelo da p-mediana capacitada:

- Restrição do número de estações abertas (*p*) : da mesma forma que no modelo da p-mediana não-capacitada, a restrição (11) foi adicionada ao modelo.

- Restrição de capacidade (6): Nos problemas da p-mediana capacitada, uma restrição de capacidade de fornecimento é atribuída às estações servidoras. Porém esta toma uma forma mais simples que no modelo de Baouche e colaboradores (2014), considerando apenas a demanda fixa, D_i, atribuída a cada agrupamento de consumo *i*:

$$\sum_{i \in I} n_i^{(ev)} Di\, y_{ij} \;\leq\; c_j x_j, \forall\, j \;\in J \qquad (13)$$

- Restrição de p-dispersão (7) ou (10): o modelo da p-mediana não conta com a restrição de p-dispersão, presente no modelo de Baouche e colaboradores (2014), devendo a mesma ser tornada sem efeito. Para isso à constante *r*, que representa a distância mínima entre duas estações de carregamento, foi atribuído o valor zero.

- f_j : Ao custo para se situar uma estação de carregamento no local candidato *j*, $\forall j \in J$, foi atribuído o valor zero, tornando o primeiro termo ($\sum_{j \in J} f_j\; x_j$) da função objetivo (3) sem efeito, já que mesmo não faz parte do modelo da p-mediana capacitada.

- $\propto$: Ao custo do quilowatt-hora, foi atribuído o valor 1, uma vez que não faz parte do modelo da p-mediana capacitada.

- $n_i^{(ev)}$: Ao número médio de veículos se deslocando para um centro de consumo *i*, $\forall i \in I$, foi atribuído o valor 1, uma vez que o mesmo não faz parte do modelo da p-mediana capacitada.

3.3.2.2 Modelo de localização modificado

O modelo da p-mediana capacitada, resultante das adequações feitas ao modelo de Baouche e colaboradores (2014) incluindo a remoção de todas as inequações e termos que foram tornados sem efeito é mostrado abaixo:

Variáveis de decisão:

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

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

Função de otimização:

$$Min \sum_{i \in I} \sum_{j \in J} d_{ij}\, y_{ij} \qquad (12)$$

Sujeito a:

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

$$y_{ij} - x_j \leq 0, \forall\, i \in I, \forall\, j \in J \quad (5)$$

$$x_j \in \{0,1\}, \forall\, j \in J, r > 0 \quad (8)$$

$$y_{ij} \in \{0,1\}, \quad \forall\, i \in I, \forall\, j \in J \quad (9)$$

$$\sum_{j=1}^{n} y_j = p\,, \forall\, j \in J \quad (11)$$

$$\sum_{i \in I} n_i^{(ev)} Di\, y_{ij} \leq c_j x_j, \forall\, j \in J \quad (13)$$

Onde:

J : conjunto de locais candidatos a receber estações de serviço;

I : conjunto de centros de consumo;

D_i : demanda de energia no centro de consumo *i*;

d_{ij} : energia necessária para um veículo se deslocar de um centro de consumo *i* a uma estação de carregamento *j;*

p : número de estações a serem abertas (alocadas);

Cj : capacidade da estação de carregamento *j*;

A função (12) representa a função de otimização. O objetivo é minimizar o custo total da demanda ponderada. A restrição (11) estabelece que o número de estações a serem abertas seja igual a *p*. A restrição(4) estabelece que toda a demanda do centro de demanda *i* deve ser satisfeita. A restrição (5) assegura que cada centro de demanda só possa ser associado a uma estação que tenha sido selecionada. A restrição (13) garante que a demanda total atribuída a estação *j* não esteja além da capacidade C_j daquela estação de serviço. Finalmente, as restrições (8) e (9) estabelecem que as variáveis de decisão devem ser binárias (0 ou 1) e não-negativas.

3.3.2.3 Bases de dados e experimentos realizados

Para avaliação do desempenho da adaptação da metaheurística CRO, foram selecionadas da literatura três bases de dados contendo problemas do tipo p-mediana capacitada (CPMP) com

tamanhos variando entre 100 e 724 estações candidatas/consumidores e número de estações selecionadas (medianas) variando entre 5 e 300. Tais bases e experimentos realizados estão relacionados a seguir:

- *Instâncias cpmp da OR-Library*: Esta base, disponibilizada na biblioteca online de problemas de pesquisa operacional *OR-Library* (BEASLEY, 1990), por Osman e Christofides (1994) e consiste de 20 problemas do tipo p-mediana capacitada. Trata-se de uma base clássica, que tem sido amplamente utilizada na literatura para *benchmarks* de soluções para CPMP, por mais de duas décadas. O primeiro grupo, contendo 10 instâncias, nomeadas de *cpmp01* a *cpmp10*, possui 50 estações candidatas / consumidores (vértices) e 5 estações selecionadas (medianas), enquanto que o último grupo, nomeado de *cpmp11* a *cmpm20*, possui 100 vértices e 10 medianas. Os resultados da presente implementação do CRO foram comparados com as seguintes metaheurísticas e métodos exatos:

  - Metaheurística hibrida, proposta por Osman e Christofides (1994), que contém elementos das metaheurísticas *Simulated Annealing* e Busca Tabu. Foi efetuada somente uma comparação dos *gaps* obtidos, uma vez que as plataformas computacionais eram muito dissimilares.

  - Resolução por otimizador comercial IBM CPLEX vs.12.6 (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017). Foram feitas comparações de *gaps* e tempo computacional com aqueles obtidos pelo CRO na mesma plataforma computacional.

  - Mateurística IRMA, proposta por Stefanello e colaboradores (2015). Foram feitas comparações de *gaps* e tempo computacional com aqueles obtidos por IRMA, rodando em hardware e software similares (Intel i5 e IBM CPLEX 12.3).

- *Instâncias sjc:* A segunda base selecionada foi proposta por Lorena e Senne (2004), e é composta de seis instâncias (*sjc1,sjc2,sjc3a,sjc3b,sjc4a e sjc4b*), construídas a partir de dados obtidos de um banco de dados de informações geográficas da cidade de São José dos Campos , SP. O número de vértices varia de 100 a 400 e o número de medianas de 10 a 40. Foram feitas comparações de *gaps* e tempo computacional, com aqueles obtidos pelos seguintes métodos:

  - Heurística *Scatter Search* (SS), proposta por Scheuerer e Wendolsky (2006).

- Heurística *Variable Neighborhood Search* (VNS) combinada com IBM CPLEX, método híbrido proposto por Fleszar e Hindi (2008).
- Heurística *Clustering Search* (CS), proposta por Chaves e colaboradores (2007).
- Técnica *Fenchel cutting planes* combinada com IBM CPLEX, denominada *Fen-CPLEX*, proposta por Boccia e colaboradores (2008).
- Mateurística IRMA, proposta por Stefanello e colaboradores (2015).

- Instâncias baseadas na TSPLIB : A última das bases selecionadas é derivada da base TSPLIB (REINELT, 1995), tendo sido modificada por Stefanello e colaboradores (2015) para o CPMP. Foram escolhidas 15 instâncias com número de vértices variando entre 318 e 724 e número de medianas entre 5 e 200. As mesmas são nomeadas de acordo com o formato <*nome da instância TSPLIB original*>_<*n*>_<*p*>, onde <*n*> corresponde ao número de vértices e <*p*>, ao número de medianas. São elas: *lin318_005, lin318_015, lin318_040, lin318_070, lin318_100, ali535_005, ali535_025, ali535_050, ali535_100, ali535_150, u724_010, u724_030, u724_075, u724_125, u724_200*.

  Cabe ressaltar que a maioria dos problemas desta base não pode ser resolvida à otimalidade por programas que utilizam métodos exatos, como o CPLEX, num tempo razoável, o que justifica o uso de heurísticas e metaheurísticas, como o CRO.

As bases de dados supracitadas, foram importadas diretamente no seu formato original ou convertidas através de programas de conversão de dados, desenvolvidos pelo próprio autor, para formatos aceitos pelo otimizador IBM CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) e pelo programa que implementa a adaptação da metaheurística CRO. Maiores informações sobre o processo de importação das bases de dados podem ser encontradas na Etapa 3.4.

Uma vez importadas e adequadas para a utilização pelo CPLEX e pelo CRO, as bases de dados selecionadas da literatura foram utilizadas com as seguintes finalidades:

- Para que pudessem ser obtidos, através do CPLEX, os valores ótimos para os problemas a serem resolvidos, bem como os tempos de execução para obtenção dos mesmos, os modelos de p-mediana, p-mediana capacitada e de Baouche e colaboradores (2014) tiveram de ser codificados na linguagem de programação, voltada a modelagem de dados, do otimizador

CPLEX, denominada *Optimization Programming Language* (OPL) (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017). Para verificar se a codificação dos modelos havia sido efetuada de forma correta e livre de erros, foi necessário confrontar os resultados obtidos pelo CPLEX com os resultados publicados na literatura, para os problemas das bases de dados selecionadas.

- Para que a eficácia da implementação da metaheurística CRO pudesse ser avaliada, o mesmo foi usado para resolver os problemas das bases de dados selecionadas, permitindo a comparação dos resultados obtidos em termos de *gaps* e, em alguns casos, de tempos computacionais, com os resultados do CPLEX e da literatura.

3.4 IMPORTAR AS BASES DE DADOS DA ETAPA 3.3 PARA USO PELA METAHEURÍSTICA CRO E POR UM OTIMIZADOR MATEMÁTICO COMERCIAL

As bases de dados selecionadas no item 3.3 foram importadas e convertidas de seu formato original para um formato aceito pelo otimizador CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) e também pelo programa de computador que implementa a adaptação da metaheurística CRO para o modelo de localização escolhido e os modelos de p-medianas capacitada e não-capacitada.

A escolha do otimizador comercial CPLEX da IBM se deve ao fato de que o mesmo tem sido amplamente utilizado na resolução de problemas de programação linear, inteira e mista, incluindo três dos métodos híbridos para solução dos problemas de p-mediana capacitada, selecionados da literatura para a avaliação da adaptação do CRO. Outra motivação relevante foi a de que a IBM permite a utilização gratuita do produto para fins acadêmicos no Brasil, bastando um simples registro e posterior validação do projeto de pesquisa e instituição de ensino. A versão disponibilizada para uso acadêmico não possui restrições quanto ao tamanho dos problemas que é capaz de trabalhar ou outras limitações.

O formato original de todas as bases de dados é o de arquivos texto. A seguir estão descritos os processos de importação das bases selecionadas da literatura para os experimentos:

3.4.1 Instâncias pmed da OR-Library

A base *pmed da OR-Library c*ontém 40 problemas de p-mediana capacitada e está disponível para download (BEASLEY, 1990). Os problemas são numerados de *pmed1* a *pmed40*, sendo armazenados em arquivos texto com o mesmo nome e extensão *.txt* (por exemplo, *pmed1.txt*).

O formato interno destes arquivos de dados é: a primeira linha contém o número de vértices, o número de arestas e *p* (número de medianas). As linhas subsequentes contêm, para cada aresta, os vértices iniciais e finais, seguidos do custo da aresta. A Figura 7 mostra as primeiras linhas do arquivo *pmed1.txt,* correspondente ao problema *pmed1*. Um pré-processamento, seguido da aplicação do algoritmo de Floyd, precisa ser aplicado aos dados contidos no arquivo, a fim de obter-se uma matriz completa que servirá para o preenchimento da matriz de energia, *d,* e da matriz de distâncias entre estações, *dist*, descritas na etapa 3.2. O pseudocódigo do pré-processamento a ser efetuado está mostrado na Figura 8.

Figura 7 – Arquivo *pmed1.txt* da biblioteca *OR-Library*

```
100 200 5
 1 2 30
 2 3 46
 3 4 1
 4 5 28
 5 6 31
 6 7 69
 7 8 39
 8 9 14
 9 10 84
 10 11 59
 11 12 10
 12 13 28
 13 14 63
 14 15 9
 15 16 100
 16 17 98
 17 18 70
 18 19 94
 19 20 22
 20 21 14
 21 22 87
 22 23 82
 23 24 55
```

Fonte: Beasley (1990)

Figura 8 – Pré-processamento a ser aplicado aos dados da biblioteca *OR-Library*

```
Let n be the number of vertices
Set c(i,j)=infinity for i=1,...,n j=1,...,n.
   c(i,i)=0 for i=1,...,n
Read each edge line in the data file IN TURN:
  if the three numbers in the line are i,j,k then
  set c(i,j)=k and c(j,i)=k
Then subject the matrix c to Floyd's algorithm to get a
symmetric allocation cost matrix.
```

Fonte: Beasley (1990)

3.4.2 Instâncias *cpmp* da *OR-Library*

Trata-se de uma base de dados disponibilizada na *OR-Library* (BEASLEY, 1990) e criada por Osman e Christofides (1994), que consiste de 20 problemas do tipo p-mediana capacitada. Existe somente um arquivo, nomeado *pmedcap.txt*, contendo todos os problemas, bem como suas soluções ótimas. O cabeçalho do arquivo consiste somente da primeira linha e contém o número de problemas, sendo que as demais linhas contêm os dados e soluções dos problemas propriamente ditos.

A primeira linha de cada problema contém o número do problema e a sua solução ótima. A linha seguinte contém o número de vértices, o número de medianas e a capacidade de cada mediana. As linhas subsequentes contêm o número do vértice, a sua localização em termos de coordenadas x e y e a demanda associada ao mesmo. Os dados armazenados estão separados pelo caractere espaço em branco. O tamanho do arquivo é de, aproximadamente, 20KB. A Figura 9 mostra as primeiras linhas do arquivo *pmedcap1.txt*, contendo o cabeçalho e parte do primeiro problema.

Para o preenchimento da matriz de energia, *d*, e da matriz de distâncias entre estações, *dist*, descritas na etapa 3.2, foi necessário efetuar um pós-processamento nos dados do problema, consistindo do cálculo das distâncias euclidianas entre as coordenadas de cada cliente, com arredondamento para o inteiro mais próximo, conforme descrito na *OR-Library* (BEASLEY, 1990).

3.4.3 Instâncias *sjc* e TSPLIB modificadas para o CPMP

As instâncias *sjc* (LORENA; SENNE, 2004) e TSPLIB, modificadas para o CPMP, foram disponibilizadas no endereço http://www-usr.inf.ufsm.br/~stefanello/instances/CPMP/ por Stefanello e colaboradores (2015). Cada instância está armazenada em um arquivo distinto, como mesmo nome da instância e extensão *.dat.* O cabeçalho do arquivo consiste apenas da primeira linha, que contém o número de vértices e o número de medianas. As linhas subsequentes contêm os dados do problema, sendo um vértice por linha. Cada vértice contém a sua localização em termos de coordenadas *x* e *y*, a capacidade do vértice (como estação servidora) e a demanda associada ao mesmo (como cliente). Os dados armazenados estão separados pelo caractere espaço em branco. A Figura 10 mostra as primeiras linhas do arquivo *sjc1.dat*, contendo o cabeçalho e parte do problema *sjc1*.

Para o preenchimento da matriz de energia, *d*, e da matriz de distâncias entre estações, *dist*, descritas na etapa 3.2, foi necessário efetuar um pós-processamento nos dados do problema, consistindo do cálculo das distâncias euclidianas entre as coordenadas de cada cliente, com arredondamento para o inteiro menor ou igual à distância calculada, conforme orientado pelo autor, Dr. Fernando Stefanello.

Figura 9 – Arquivo *pmedcap1.txt* da biblioteca *OR-Library*

```
20
 1 713
 50 5 120
 1 2 62 3
 2 80 25 14
 3 36 88 1
 4 57 23 14
 5 33 17 19
 6 76 43 2
 7 77 85 14
 8 94 6 6
 9 89 11 7
 10 59 72 6
```

Fonte: Beasley (1990)

Um programa de computador, codificado em linguagem de programação C# (EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION, 2006), foi criado e utilizado para ler os arquivos das bases de dados capacitadas e não-capacitadas, efetuar as transformações necessárias e prover, como saída, arquivos num formato que pudessem ser processados pelo otimizador CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017).

A saída de dados também foi efetuada na forma de instâncias de classe, contendo estruturas em memória (vetores e listas encadeadas) que serviram como entrada para um outro programa, também desenvolvido em linguagem C# (EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION, 2006), que implementou a metaheurística CRO (LAM; LI, 2012) para os modelos de otimização utilizados nesta dissertação.

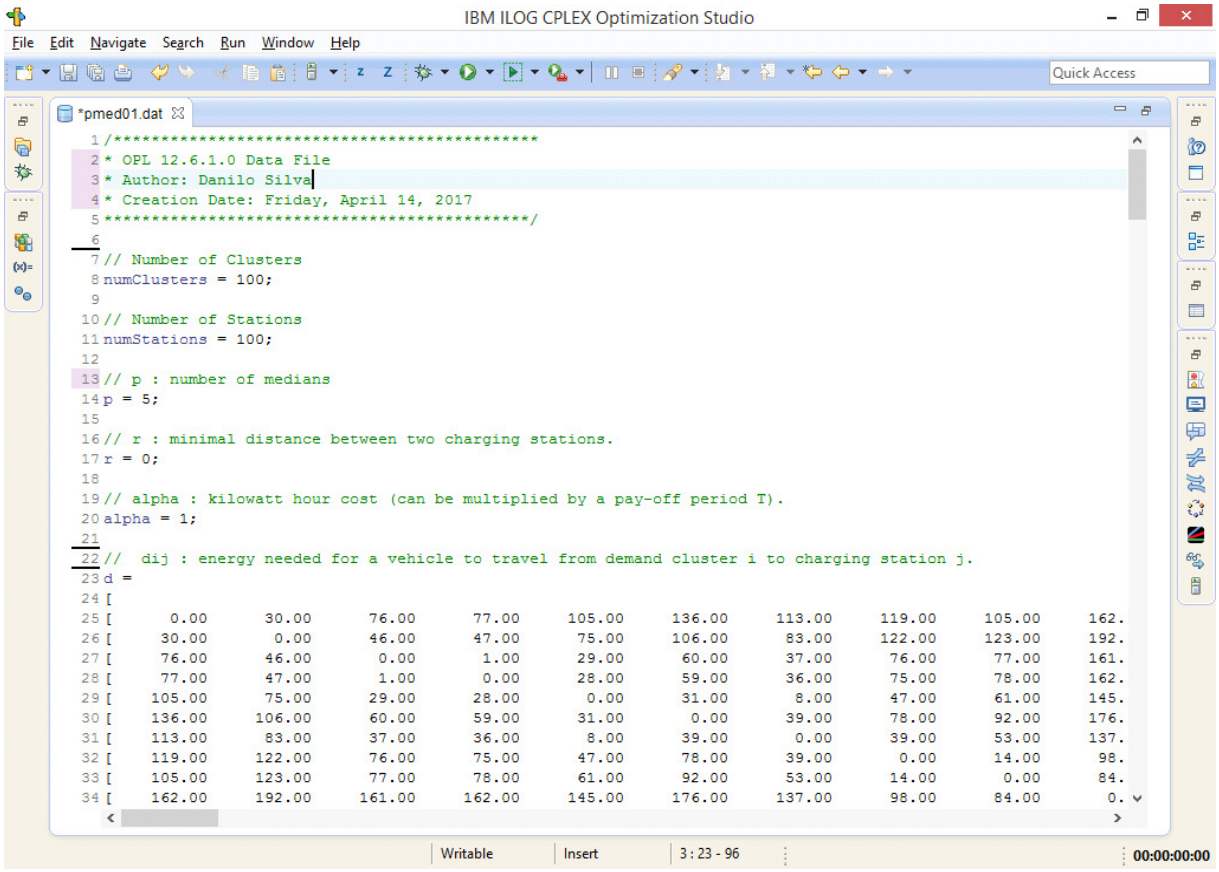
Figura 10 – Arquivo *SJC1.dat*, contendo o problema *sjc1*

```
100 10
  409154  435528     720      50
  409151  435683     720       4
  409277  435420     720      33
  409260  435538     720      15
  409240  435695     720       1
  409213  435897     720       5
  409199  435982     720      87
  409178  436078     720      91
  409174  436171     720      45
  409147  436256     720       1
  409469  435389     720      41
  409378  435463     720      23
  409378  435563     720      26
  409351  435734     720     301
  409328  435891     720      53
  409275  435982     720       1
```

Fonte: Stefanello e colaboradores (2015)

A Figura 11 mostra um dos arquivos (*pmed01.dat*) criados pelo programa de conversão, contendo o primeiro problema de p-mediana capacitada (*pmed1*), já aberto pelo CPLEX e pronto para ser processado.

Figura 11 – Arquivo de entrada do CPLEX, contendo problema *pmed01*



Fonte: Elaborado pelo autor (2017)

3.5 CODIFICAR O PROBLEMA DE OTIMIZAÇÃO DA ETAPA 3.2, NA LINGUAGEM DE PROGRAMAÇÃO DO OTIMIZADOR MATEMÁTICO COMERCIAL

O otimizador IBM CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) foi utilizado para a resolução do modelo de otimização para localizações de estações de carregamento de EVs, descrito na etapa 3.2. Os valores e tempos de execução obtidos foram utilizados para avaliar a qualidade da adaptação da metaheurística CRO ao problema de localização adotado. O CPLEX também foi utilizado para a resolução das bases de dados selecionadas da literatura, que empregam modelos de p-mediana, com a mesma finalidade.

O IBM CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) é um produto de software oferecido pela IBM para modelagem e resolução de modelos matemáticos de otimização. O produto oferece uma plataforma flexível e de alto desempenho para resolução de problemas de programação linear, inteira, mista e quadrática. Uma descrição detalhada do produto pode ser obtida em International Business Machines Corporation (2017).

Para que o modelo de otimização adotado, bem como modelos de p-mediana, pudessem ser processados pelo CPLEX, foi necessário a codificação dos mesmos em linguagem de modelagem OPL (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017), a qual faz parte do otimizador CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017). A codificação das variáveis de entrada e decisão, função objetivo e modelo proposto em linguagem OPL, para o modelo de Baouche e colaboradores (2014), pode ser vista na Figura 12.

As bases de dados selecionadas da *OR-Library* (*pmed* e *cpmp*) foram utilizadas para verificar se a codificação em OPL dos modelos de p-mediana capacitada, não capacitada e modelo de Baouche e colaboradores (2014) foi efetuada corretamente. Para os dois primeiros modelos a verificação foi feita comparando-se os valores obtidos pelo CPLEX com os da literatura, que deveriam ser idênticos para que os modelos OPL estivessem corretos, uma vez o otimizador utiliza métodos exatos para resolução dos modelos, sempre buscando valores ótimos.

O modelo de Baouche e colaboradores (2014) foi codificado por último, já que é mais complexo e possui diversas restrições comuns aos modelos de p-mediana. Uma vez que valores ótimos não estavam disponíveis na literatura, uma análise foi efetuada, resolvendo o modelo de Baouche e colaboradores (2014) para todos os problemas da *OR-Library,* atribuindo-se uma distância mínima entre estações, *r*, e comparando-se o número de estações abertas obtido com os dos modelos de p-mediana, cujo o número de estações (medianas) é fixo. Uma descrição mais completa dos testes efetuados, incluindo alguns dos resultados obtidos pode ser encontrada no APÊNDICE A.

Adicionalmente à codificação do modelo proposto, em linguagem OPL, foram implementados, em linguagem de programação *JavaScript* (EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION, 2009), scripts para processamento em lote de todos os problemas de p-mediana (capacitada e não-capacitada), com geração de saída de resultados em arquivo em disco, para posterior análise. Os dados armazenados incluem uma identificação do programa utilizado para resolução do modelo, a identificação do problema na base de dados, o número de medianas, o número de estações, o número de agrupamentos de demanda, o valor obtido para a função objetivo, o tempo de processamento, além de uma lista de medianas (estações de carregamento) selecionadas. Uma saída típica em arquivo do programa em *JavaScript* para execução em lote dos problemas da p-mediana pode ser vista na Figura 13.

Figura 12 – Implementação em linguagem OPL do modelo FCLP Baouche

```
/*********************************************
 * OPL 12.6.1.0 Model
 * Author: Danilo A. Silva
 * Creation Date: Apr 17, 2016 at 6:59:56 AM
 *********************************************/
int numClusters = ...; range clusters = 1..numClusters; // Number of Clusters
int numStations = ...; range stations = 1..numStations; // Number of Stations
int r = ...; // r : minimal distance between two charging stations

// Proposed model (BAOUCHE et al.,2014)
float f [stations] = ...; // fj : cost of locating a charging station in candidate site j
float d [clusters][stations] = ... ;  // dij : energy needed for a vehicle to travel from demand cluster i to station j
float ed [clusters] = ...; //Di : energy demand at cluster i
float dist[stations][stations] = ...; // distij : distance between charging station i and charging station j
float c [stations] = ...; // Cj : capacity of charging station j
float alpha = ...; // alpha : kilowatt hour cost (can be multiplied by a pay-off period T)
float nev [clusters] = ...; // nevi : number of EVs traveling to zone/cluster i
float M =  2 * max (i in stations, j in stations) dist[i][j]; // M constant

// Decision variables
dvar int y [clusters][stations] in (0..1); // 1 if demand cluster is i covered by the station j, otherwise 0
dvar int x [stations] in (0..1);  // 1 if we locate a station in candidate site j, otherwise 0

// Objetive: Minimize the total cost and total distance traveled by all vehicles to access the selected stations
minimize   sum (j in stations ) f[j] * x[j] + alpha * sum (i in clusters, j in stations) nev[i] * d[i][j] * y [i][j];
subject to {
    // Demand cluster is covered by one charging station
    forall (i in clusters)ct3: sum (j in stations) y[i][j] == 1;
    forall (i in clusters, j in stations) ct4: y[i][j] - x[j] <= 0;
    // Demand assigned at location j is not beyond the capacity Cj of that charging station
    forall (j in stations) ct5: sum (i in clusters) nev[i] * ( ed[i] + d[i][j] ) * y[i][j] <= c[j] * x [j];
    // Each server must be separated with a minimum radius r
    forall (i in stations, j in stations : i+1 <= j)
              ct9a: ( r + ( M - dist[i][j] )* x[i] + ( M - dist[i][j]) * x[j] <=  2 * M - dist[i][j]);
    r >= 0; // r must be positive  }
```

Fonte: Elaborado pelo autor (2017)

Figura 13 – Saída da execução em lote de problemas no CPLEX

```
Alg,ProblemID,P,NumStations,NumClusters,Objective,Duration,ST0,ST1,ST2,ST3,ST4,ST5,ST6,ST7,ST8,ST9,ST10,ST11,...
CPLEX,pmed01,5,100,100,5819,360,6,12,64,90,98
CPLEX,pmed02,10,100,100,4093,609,5,7,11,36,40,44,66,90,94,98
CPLEX,pmed03,10,100,100,4250,453,4,8,12,20,25,35,47,54,68,98
CPLEX,pmed04,20,100,100,3034,266,5,6,8,12,21,25,33,37,49,54,59,65,71,76,82,86,90,92,95,99
CPLEX,pmed05,33,100,100,1355,297,3,6,8,13,18,24,25,28,29,32,35,36,37,40,48,50,52,55,57,65,68,69,72,74,80,81,...
CPLEX,pmed06,5,200,200,7824,5813,15,85,100,110,125
CPLEX,pmed07,10,200,200,5631,2859,2,9,71,86,115,130,141,180,185,190
CPLEX,pmed08,20,200,200,4445,2766,41,65,69,75,82,95,103,113,116,118,126,129,132,138,145,153,166,179,193,198
CPLEX,pmed09,40,200,200,2734,2609,0,2,11,18,24,28,30,39,47,53,54,60,66,69,71,76,88,90,95,97,100,107,121,125,...
CPLEX,pmed10,67,200,200,1255,2406,2,11,16,18,30,34,38,40,41,42,46,54,57,58,63,64,65,67,68,69,74,75,79,80,81,84,...
CPLEX,pmed11,5,300,300,7696,24281,23,30,97,166,200
CPLEX,pmed12,10,300,300,6634,19031,2,5,43,83,138,168,171,173,216,293
CPLEX,pmed13,30,300,300,4374,13468,1,5,12,16,26,29,38,40,48,54,62,75,83,92,105,107,123,131,142,146,150,160,...
CPLEX,pmed14,60,300,300,2968,14547,0,6,12,13,14,17,21,22,35,48,49,50,57,60,63,71,76,78,79,82,101,106,110,...
CPLEX,pmed15,100,300,300,1729,12719,0,3,4,7,9,10,14,16,17,20,22,26,28,31,36,37,42,44,45,48,50,58,60,61,65,72,...
CPLEX,p-medcap01,5,50,50,713,265,9,11,18,20,47
CPLEX,p-medcap02,5,50,50,740,109,15,21,25,32,46
CPLEX,p-medcap03,5,50,50,751,172,14,19,37,38,47
CPLEX,p-medcap04,5,50,50,651,140,2,8,28,42,49
CPLEX,p-medcap05,5,50,50,664,203,12,21,28,35,39
CPLEX,p-medcap06,5,50,50,778,203,6,16,19,41,45
CPLEX,p-medcap07,5,50,50,787,578,5,12,19,23,35
CPLEX,p-medcap08,5,50,50,820,2625,1,15,24,29,39
CPLEX,p-medcap09,5,50,50,715,360,0,6,10,21,37
CPLEX,p-medcap10,5,50,50,829,2360,5,15,33,40,49
CPLEX,p-medcap11,10,100,100,1006,3266,6,21,44,51,62,68,73,74,79,99
CPLEX,p-medcap12,10,100,100,966,2782,1,12,16,31,59,65,66,76,80,95
CPLEX,p-medcap13,10,100,100,1026,953,16,35,50,53,58,63,73,74,78,81
CPLEX,p-medcap14,10,100,100,982,4562,2,5,22,24,34,36,49,91,94,98
CPLEX,p-medcap15,10,100,100,1091,4812,4,7,21,44,52,61,84,87,91,95
```

Fonte: Elaborado pelo autor (2017)

3.6 ADAPTAR A METAHEURÍSTICA CRO PARA A RESOLUÇÃO DO PROBLEMA DA ETAPA 3.2, ALÉM DOS PROBLEMAS DAS BASES DE DADOS SELECIONADAS NA ETAPA 3.3

No presente trabalho a metaheurística CRO foi implementada na forma de aplicação codificada na linguagem de programação orientada a objetos C# (EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION, 2006), dada a facilidade de modelagem das moléculas como instâncias de uma classe que contém todos os atributos necessários ao seu funcionamento. As reações elementares foram implementadas na forma de métodos de uma classe que contém os códigos dos diferentes algoritmos utilizados nas reações.

No itens a seguir estão descritos os detalhes da presente implementação do CRO para o problemas da p-mediana capacitada (CPMP) e o modelo de localização de Baouche e colaboradores (2014).

O desenvolvimento do CRO para o modelo da p-mediana clássica ou não-capacitada, requereu estruturas de dados e operadores mais simples, e está descrito no APÊNDICE B. Os resultados computacionais para a p-mediana clássica são apresentados na seção 4.

Maiores informações sobre a metaheurística CRO podem ser obtidas em Lam e Li (2012).

3.6.1 Molécula

A molécula é a unidade básica do CRO, e contém vários atributos requeridos para as operações do CRO. Para a presente implementação, destinada a resolução dos problemas de p-mediana capacitada e FCLP Baouche (BAOUCHE et al., 2014), os seguintes atributos foram definidos:

- **Identificador da Molécula (*MolID*):** identifica de forma inequívoca, uma molécula na população de moléculas.

- **Estrutura molecular (ω):** armazena uma solução viável para o problema, ou seja, o valor da função objetivo, além das variáveis de decisão x (1) e y (2).O conjunto de variáveis de decisão x armazena as estações correntemente selecionadas de uma determinada solução viável. Foram implementadas como uma lista de inteiros, armazenando apenas os números das estações que fazem parte da solução. Por exemplo, se o número de estações candidatas for 100 e o número de medianas for cinco, x armazenará cinco valores correspondentes às estações atualmente selecionadas, como na lista {5, 17, 29, 45 e 79}. Isso provou ser mais

eficaz do que armazenar *x* como uma matriz de bits de tamanho *J*. Da mesma forma, o conjunto de variáveis de decisão *y*, que armazena as atribuições cliente-estação, foi implementada como uma matriz de inteiros de tamanho *I*, em vez de uma matriz de bits de tamanho *J x I*.

- **Energia potencial (*PE*):** é definida como o valor da função objetivo da solução correspondente, representada por ω. Se *f* denota a função objetivo, então tem-se que: $PE\omega = f(\omega)$.

- **Energia cinética (*KE*):** é um número não negativo que quantifica a tolerância do sistema em aceitar uma solução pior do que a existente.

- **Número de colisões (*NumHit*):** número total de colisões que uma molécula já sofreu.

- **Estrutura mínima (*MinStruct*):** é a solução (ω) com energia potencial (*PE*) mínima, que uma molécula atingiu, até um dado momento. Preserva a estrutura com o menor *PE* no histórico de reações.

- **Energia potencial mínima (*MinPE*):** é o *PE* correspondente ao *MinStruct*.

- **Número mínimo de colisões (*MinHit*):** é o número da colisão em que a molécula atingiu o menor valor de *PE* (*MinPE*).

- ***Conjunto de Listas de Proximidade (CLP)*:** definido como um conjunto de tamanho *J*. Cada elemento *CLPj* do conjunto, contém uma lista de vértices (clientes/estações) que estão próximas ao vértice *j*. Estas listas são construídas utilizando uma estratégia originalmente apresentada por Stefanello e colaboradores (2015), para sua heurística denominada "*Iterated Reduction Matheuristic Algorithm*" (IRMA) R1. A estratégia original foi modificada para um prover um conjunto de listas contendo estações candidatas próximas às estações correntemente selecionadas em uma dada solução, e que poderiam substituir as mesmas. Tais listas são utilizadas durante as fases de intensificação do CRO (colisões unimoleculares e intermoleculares, com o propósito de reduzir a quantidade total de iterações requeridas pelo mecanismo λ-*interchange* (OSMAN; CHRISTOFIDES, 1994). O processo de construção do CLP é baseado em capacidade e mostrado em detalhes no item 3.6.4.

κ* :** é um parâmetro usado como um fator de expansão de capacidade para controlar o número de estações próximas armazenados em cada uma das listas do conjunto de listas de proximidade (CLP). Se *f* denota uma função que gera um CLP, contendo listas de estações próximas à cada estação candidata, então: ***CLPκ = f (κ).

O pseudocódigo para a classe molécula (*Molecule*) é mostrado na Figura 14. Ele contém somente propriedades, além de um construtor. O código do construtor é chamado por um algoritmo de construção, responsável por gerar um número aleatório de soluções viáveis. O mesmo recebe um Identificador da Molécula (***MolID***), uma solução viável e uma quantidade inicial de energia cinética (***InitialKE***). No código do construtor, um novo Conjunto de Listas de Proximidade (CLP), com fator de capacidade κ_0, é criada.

Figura 14 – A Classe Molécula

```
class Molecule
  Attributes:
    MolID, ω, PE, KE, NumHit, MinStruct, MinPE, MinHit, CLP and κ
  Methods:
 // Constructor
 Molecule (molID, initialSolution, initialKE ) \\constructor
 {
    MolID ← molID;
    ω ← initialSolution;
    PE ← f (ω);
    KE ← initialKE;
    NumHit ← 0;
    MinStruct ← ω;
    MinPE ← PE;
    MinHit ← 0;

    // k0 :constant that sets the min. CLP capacity factor for all molecules
    CLP   ← new CLP (κ0); // Build new CLP
}
End class
```

Fonte: Elaborado pelo autor (2017)

3.6.2 Inicialização e fase construtiva

A fase de inicialização consiste em atribuir valores apropriados aos parâmetros operacionais do CRO, conforme definidos em Lam e Li (2012). Tais parâmetros são ***PopSize, KELossRate, MoleColl, buffer, InitialKE, α*** e ***β.*** Uma breve descrição dos mesmos é dada a seguir:

- ***PopSize:*** é um número inteiro que denota o tamanho da população inicial de moléculas. Devido ao tamanho relativamente grande de muitas das instâncias avaliadas nesta dissertação, populações iniciais pequenas, com tamanho entre 2 e 10 moléculas foram utilizadas, de modo a reduzir os tempos de inicialização.

- ***KELossRate:*** usado pelo CRO durante colisões ineficazes com a parede para determinar a quantidade mínima de energia cinética que a molécula reterá de sua energia inicial, após uma colisão. O valor 0,8 foi atribuído para *KELossRate* em todas as instâncias de teste.

- ***MoleColl*****:** um valor real que denota a probabilidade de uma colisão intermolecular ocorrer. O valor 0,1 foi atribuído para ***MoleColl*** em todas as instâncias de teste.

- ***Buffer*****:** reservatório de energia central do CRO. Inicializado com zero em todas as instâncias de teste.

- ***InitialKE*****:** denota a quantidade de energia cinética dada a uma nova molécula.

- ***α*****:** define o número máximo de vezes que uma molécula pode fazer uma busca local sem que um melhor mínimo local seja encontrado, antes que a mesma ser torne elegível para uma decomposição.

- ***β***: moléculas com um *KE* muito baixo perdem a sua flexibilidade em escapar de um mínimo local. β define a energia mínima que a molécula necessita atingir para que a mesma se qualifique para uma síntese.

O algoritmo construtivo cria uma nova população de moléculas contendo soluções viáveis, i.e., soluções que não violem as restrições do modelo CPMP ou FCLP Baouche, dependendo do modelo que o CRO esteja utilizando no momento. Um processo composto de 6 etapas é utilizado na construção de uma nova solução, e se baseia numa heurística primal proposta por Mulvey e Beck (1984), um algoritmo de busca em vizinhança proposto por Maranzana (1964) e o algoritmo *Fast Interchange*, proposto por Whitaker (1983). O processo é delineado a seguir:

a. Uma população preliminar de moléculas 100 vezes maior que a população desejada (*PopSize*) é criada. Em seguida, para cada uma das soluções criadas, *p* estações, ou medianas, são escolhidas aleatoriamente do conjunto de candidatas. O número de medianas, *p*, é fixo no modelo CPMP ou gerado aleatoriamente no modelo FCLP Baouche. Uma heurística primal proposta por Mulvey e Beck (1984) é, então, aplicada a todas as soluções, da seguinte forma: clientes são atribuídos às medianas selecionadas em ordem decrescente de seu valor "*regret*". *Regret* é definido como o valor absoluto da diferença, em termos de distância (ou energia), entre a primeira e a segunda mediana mais próxima ao cliente.

b. Uma vez que o item *a* esteja completado, a população de moléculas é reduzida ao número especificado em *PopSize*. São selecionadas as moléculas que contêm os menores valores de função objetivo dentre a população preliminar. As moléculas restantes são descartadas. Este procedimento, que não requer um grande esforço computacional, provou ser eficaz no aumento da qualidade das soluções. Em seguida os passos *c* a *f* são executados para cada uma das soluções (ω) armazenadas nas moléculas selecionadas.

c. Com base na premissa de que um conjunto de medianas selecionadas para obter-se uma boa solução do problema de p-mediana capacitada deve estar próximo a um conjunto de medianas selecionadas para uma boa solução de p-mediana não-capacitada, com respeito à distância entre esses conjuntos, as soluções criadas na etapa b são otimizadas, ignorando-se as restrições de capacidade (6) ou (13), dependendo do modelo. Nesta etapa, busca-se o melhor mínimo local não-capacitado, usando o algoritmo *Fast Interchange*, proposto por Whitaker (1983) e posteriormente modificado por Hansen e Mladenovic (1997). Este algoritmo é explicado, de forma sucinta, na seção 3.6.3.

d. Uma vez encontrado um mínimo local, a mesma heurística primal proposta por Mulvey e Beck (1984) e descrita no item *a* é aplicada a solução não-capacitada, a fim de obter-se uma solução capacitada.

e. Em seguida, o algoritmo construtivo examina cada um dos vértices clientes *i* atribuídos a uma dada uma estação (mediana) *j*, buscando encontrar uma outra estação candidata *j'* que minimize o custo total ponderado pela demanda *Di,* no modelo CPMP, ou $n_i^{(ev)}(Di + d_{ij'})$, no modelo FCLP Baouche, dentre todos os vértices de um conjunto

particular. Se um vértice melhor do que o vértice já selecionado para ser uma estação (mediana) for encontrado, este se tornará a nova mediana e todos os outros vértices desse conjunto serão reatribuídos a ele. Neste ponto, pode ocorrer que a restrição de capacidade (6) de FCLP Baouche ou (13) do CPMP seja violada, se a mediana recém-descoberta não tiver capacidade suficiente para atender a todos os vértices do conjunto. Caso isto ocorra, a mudança de mediana será descartada. Este procedimento foi proposto por Maranzana (1964) como parte de seu algoritmo de busca de vizinhança.

f. Uma vez que o passo *e* esteja terminado, os vértices cliente são novamente atribuídos às suas medianas em ordem decrescente de seu valor *regret*, conforme descrito no item *a*. O passo *e* é repetido até que um limite de 20 iterações sem melhorias seja atingido. A melhor solução obtida é preservada e se torna a solução inicial da molécula (ω).

Além de obter soluções viáveis para todas as moléculas, um novo Conjunto de Listas de Proximidade (CLP), sendo um conjunto para cada molécula, é criado pelo algoritmo construtivo. O processo para geração de um CLP é descrito na seção 3.6.4.

3.6.3 O algoritmo Fast Interchange

Nas colisões ineficazes uni e multimoleculares um operador baseado no algoritmo *Fast Interchange* proposto por Whitaker (1983) é utilizado. No trabalho de Whitaker, três ingredientes eficientes são incorporados na heurística de troca padrão:

- Avaliação de movimento (*Move evaluation*), onde a melhor estação a ser removida é encontrada quando a estação a ser adicionada é conhecida. O pseudocódigo do procedimento *Move* é apresentado na Figura 42.

- Atualização da primeira e da segunda estação mais próxima de cada centro de consumo (*Update*). O pseudocódigo do procedimento *Update* é apresentado na Figura 43.

- Estratégia restrita a primeira melhoria (*first improvement*), onde cada estação é considerada para ser adicionada apenas uma vez. Hansen e Mladenovic (1997) implementaram uma versão modificada do algoritmo *Fast Interchange* para ser usada na metaheurística *Variable Neighborhood Search* (VNS), na qual o mesmo foi modificado para uma estratégia de obtenção da maior melhoria (*best improvement*). Na presente implementação, o algoritmo modificado por Hansen e Mladenovic (1997) foi utilizado, sendo também possível

configurá-lo, através de um parâmetro de entrada, para terminar assim que a primeira melhoria houver sido obtida, ou executá-lo até que a maior melhoria tenha sido alcançada. O pseudocódigo do algoritmo *Fast Interchange* é apresentado na Figura 44.

Inicialmente, o algoritmo atribui os clientes mais próximos à cada estação, relaxando assim a restrição de capacidade (6) do modelo FCLP Baouche ou (13) do modelo CPMP. Em seguida, o algoritmo *Fast Interchange* é usado para melhorar a solução não-capacitada, a fim de obter a maior melhoria. Finalmente, a heurística primal de Mulvey e Beck (1984) é aplicada à solução incumbente, conforme descrito no item *a* do item 3.6.2.

3.6.4 O Conjunto de Listas de Proximidade

O Conjunto de Listas de Proximidade (CLP) é uma estrutura de dados usada para limitar o número de iterações realizadas pelo mecanismo *λ-interchange*, proposto por Osman e Christofides (1994), executado durante as fases de intensificação do CRO. O mecanismo *λ-interchange* é descrito no item 3.6.5. Devido ao tamanho de algumas das instâncias avaliadas, tornou-se necessário empregar uma estratégia para reduzir o número de operações de troca e deslocamento que são típicas do algoritmo *λ-interchange*. Isso é especialmente importante nos casos em que *p* (número de medianas) é alto.

Em um mecanismo *λ-interchange* tradicional (OSMAN; CHRISTOFIDES, 1994), cada cliente é sistematicamente reconectado a todas as estações selecionadas (medianas), desde que sejam diferentes daquelas a que estão atualmente conectados, uma estação por vez, com o intuito de buscar possíveis melhorias no valor da função objetivo.

Na versão modificada para o CRO, considera-se, para fins de reconexão, apenas as estações selecionadas que estão nas proximidades da estação à qual um cliente está atualmente conectado. Para conseguir isso, constrói-se para cada vértice uma lista estática contendo vértices que estão próximos daquele vértice em particular. A esta lista dá-se o nome de Lista de Proximidade (LP).

O Conjunto de Listas de Proximidade (*CLP*) é um conjunto de *LP*s de tamanho *J*. Cada elemento LP_j do conjunto contém uma lista de vértices (clientes/estações) que estão próximos a estação *j*. O método usado para o preenchimento das LP foi adaptado de uma heurística proposta por Stefanello e colaboradores (2015), que foi originalmente projetada para efetuar a redução de modelos matemáticos de otimização, através da eliminação de variáveis de decisão que são improváveis de pertencer a boas soluções.

Considere a variável de decisão x_j. O subconjunto $LP_j \subseteq J$, dos vértices próximos de j é definido como:

$$LP_j = \{t \in J \mid \Sigma D_t \leq \kappa C_j - D_j\} \quad (14)$$

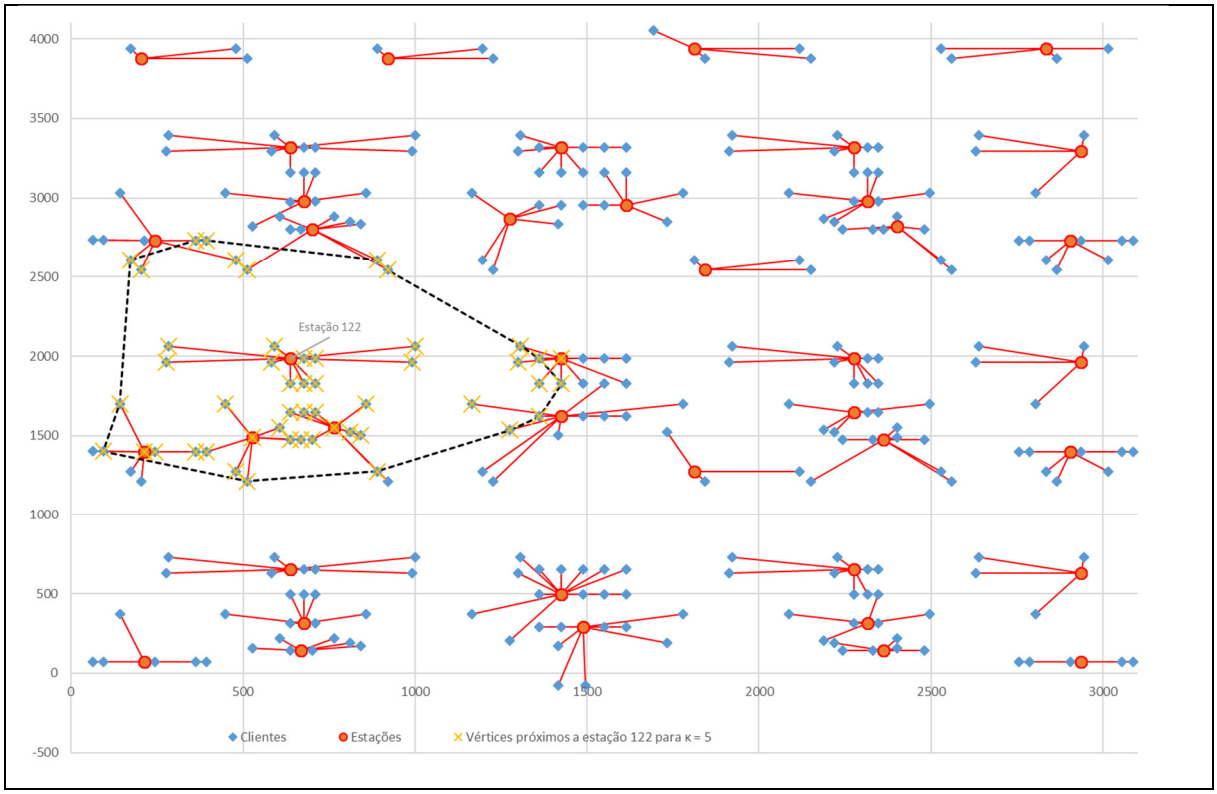
Onde um vértice t está mais próximo a j do que um vértice t' se $d_{jt} < d_{jt'}$. Assim, para uma dada mediana candidata $j \in J$, inclui-se a variável x_t na sua lista de proximidade se $t \in LPj$. O parâmetro κ é um fator de expansão de capacidade usado no controle do tamanho das listas de proximidade. A Figura 15 mostra uma solução viável para uma dos problemas (*lin318_040*) avaliados nesta dissertação. Os vértices que são parte da Lista de Proximidade da estação 122 (LP_{122}), para $\kappa = 5$, estão identificados com uma cruz. A linha tracejada demarca os limites da Lista de Proximidade LP_{122}, ou seja, os vértices mais distantes da estação 122. Para a solução viável mostrada na Figura 15, três estações seriam consideradas pelo mecanismo *λ-interchange*, para fins de reconexão dos clientes conectados a estação 122. Um valor mais alto para κ, permitiria incluir outras estações um pouco mais distantes, e vice-versa.

O pseudocódigo parcial para a classe CLP que armazena um Conjunto de Listas de Proximidade é mostrada na Figura 15. O construtor da classe recebe somente um parâmetro, κ, que controla indiretamente o tamanho das listas pertencentes ao conjunto. Para cada estação candidata (vértice) *j,* constrói-se uma lista de arestas contendo, para cada vértice $i \mid i \neq j$:

- A distância (ou energia, dependendo do modelo) de i para j (d_{ij}).
- A demanda D_i, associada ao vértice i.

Então, a lista de arestas resultante é ordenada por d_{ij}, em ordem crescente. Por fim, a Lista de Proximidade é preenchida, selecionando-se os primeiros t elementos que satisfazem (14).

Figura 15 – Lista de proximidade para a estação 122 do problema lin318_040



Fonte: Elaborado pelo autor (2017)

Figura 16 – A classe *CLP*

```
class CLP
Attributes:
   LP: List of List of integer and κ
Variables:
   edge: Structure // Struct to track node X median assignments for further sorting
          i, d, D
         EndStructure
Methods:
CLP (κ) \\constructor
{
    // Build proximity lists for all nodes
   for (j ← 0; j < J; j++)
   {
         // Create new LPj list
         LP[j] ← new List of integer;

         // Build list of edges and sort it by distance dij
         edgeList = new List of edge;
         for (i ← 0; i < I; i++) If (i <> j) then edgeList.Add (i,d[i],D[i]);
        edgeList.SortbyDistanceAscending (d);

         // Find nearest nodes of j
         s ← 0; l ← 0;
         while ((s ≤ ((κ * c[j]) – D[j])) and (l < edgeList.Count)) {
            s ← s + edgeList[l].D;
            l++;
         }
          // Build proximity list for LPj
         for (t ← 0; t < l-1; t++)
      LP[j].Add(edgeList[t].i);
   }
}
End class
```

Fonte: Elaborado pelo autor (2017)

3.6.5 O mecanismo *λ-interchange* de busca em vizinhança

Em todas as colisões ineficazes, seja contra a parede ou intermoleculares, foi implementado um operador de busca em vizinhança baseado no mecanismo *λ-interchange*, proposto Osman e Christofides (1994) para o *Capacitated Clustering Problem* (CCP), que é uma outra denominação para o CPMP. O mecanismo *λ-interchange*, por sua vez, é uma adaptação de um mecanismo de geração denominado "*λ-opt procedure*", baseado no algoritmo *arcs-exchange*, proposto por Lin (1965), para o problema do caixeiro viajante (TSP, do inglês *Traveling Salesman Problem*).

O mecanismo *λ-interchange* gera novas vizinhanças, como se segue: Seja C_i um agrupamento composto por um número de clientes conectados a uma estação de serviço (mediana), ζ_i. Dada uma solução $S = \{C_1,\ldots, C_i,.., C_j,\ldots, C_p\}$ com ζ_i sendo a mediana do agrupamento C_i, uma operação de troca (*λ-interchange*) entre dois agrupamentos C_i e C_j é a substituição do subconjunto $\bar{C}_i \subseteq C_i$, de tamanho $|\bar{C}_i| \leq \lambda$, por outro subconjunto $\bar{C}_j \subseteq C_j$, de tamanho $|\bar{C}_j| \leq \lambda$, para obter-se dois novos agrupamentos $C'_i = (C_i - \bar{C}_i) \cup \bar{C}_j$ e $C'_j = (C_j - \bar{C}_j) \cup \bar{C}_i$, com possivelmente duas novas medianas ζ'_i e ζ'_j, respectivamente. A nova solução então se torna $S' = \{C_1,\ldots,C'_i,..,C'_j,\ldots,C_p\}$. A vizinhança $N(S)$ de S é o conjunto de todas as soluções S' geradas pelo mecanismo *λ-interchange* para um dado valor inteiro λ e é denotado por $N_\lambda(S)$.

Seja LP_i uma lista de proximidade contendo vértices próximos a ζ_i (a mediana do agrupamento C_i) e m_i o número de vértices em LP_i que também sejam medianas de outros agrupamentos. O mecanismo *λ-interchange* somente irá examinar os pares de agrupamentos (C_i, C_j) onde $\zeta_j \in LP_i$. Desta forma, para um dado λ, o número de pares de agrupamentos a serem examinados é dado por m_i.

Para qualquer par de agrupamentos (C_i, C_j), o mecanismo *λ-interchange* utiliza dois processos para gerar vizinhanças. Seja $\mu \mid 1 \leq \mu \leq \lambda$, o número de clientes de C_i ou C_j a serem examinados, para fins de reconexão com outras estações (medianas), por qualquer um dos dois processos:

- Um processo de deslocamento (*shift*) tenta mover μ clientes de C_i para C_j, ou vice-versa. Para $\mu = 1$, um processo de deslocamento é representado pelos operadores *(0,1)* e *(1,0)*.

- Um processo de troca (*interchange*), como o nome sugere, tenta trocar até μ clientes do primeiro agrupamento com até μ clientes do segundo agrupamento, e vice-versa. Para $\mu = 1$, o processo é representado pelo operador *(1, 1)*.

A Figura 17 ilustra os processos de deslocamento e troca para um mecanismo *λ-interchange* cujo $\lambda = 1$, também denominado *1-interchange*. Um processo de deslocamento ocorre da Figura 17 (a) para a Figura **17** (b), onde um cliente *i* é deslocado pelo operador *(1,0)*. Como resultado, o cliente *j* se torna a nova mediana (estação servidora). As Figura 17(c) e (d), mostram as mudanças nos agrupamentos após os clientes *i* e *j* serem trocados, o que também causa mudança nas medianas em ambos os agrupamentos.

Na presente implementação, uma ordem de busca diferente da empregada por Osman e Christofides (1994) foi utilizada. Primeiramente, tenta-se trocar clientes entre medianas, antes de tentar deslocá-los. Tal estratégia provou ser mais efetiva nos testes realizados. Assim a ordem de operações, para $\lambda = 1$, se torna *(1,1), (1,0) e (0,1).* Para $\lambda = 2$, a ordem utilizada foi *(1,1), (1,0), (0,1), (1,2), (0,2), (2,1), (2,0), (2,2).*

A implementação do mecanismo *λ-interchange* para a presente adaptação do CRO é descrita a seguir:

Inicialmente, um Conjunto de Listas de Proximidade (CLP) é criado para um valor inicial de κ (κ_0). Então, a partir de uma solução viável *S*, a lógica do *λ-interchange* é executada por um número específico de iterações, numa tentativa de melhorar a solução *S*, ou seja, reduzir o valor da função objetivo. O processo iterativo do *λ-interchange* é mostrado abaixo:

Seja $\mu \mid 1 \leq \mu \leq \lambda$, o número de clientes do agrupamento C_i a serem considerados para operações de deslocamento ou troca e $\mu' \mid 1 \leq \mu' \leq \lambda$, o número de clientes do agrupamento C_j a serem considerados para operações de deslocamento ou troca. Durante uma única iteração, começando com $\mu = 1$ e $\mu' = 1$, uma busca examina, de forma randômica, todo os pares de agrupamentos (C_i, C_j) possíveis, sem repetições, procurando por conjuntos de μ clientes de C_i e μ' clientes de C_j, que possam ser deslocados ou trocados e que causem uma melhoria na função objetivo.

Para determinar um par de agrupamentos (C_i, C_j,), uma mediana ζ_i é randomicamente selecionada de uma das medianas que pertencem a solução *S*. Então, uma outra mediana $\zeta_j \mid \zeta_j \in LP_i$, é também randomicamente selecionada de *S*. Se uma operação de deslocamento ou troca melhora o valor da função objetivo, (3) ou (12), e não viola nenhuma das restrições, a operação é executada imediatamente e uma nova mediana é recomputada para os agrupamentos afetados pela operação. De modo a manter os tempos computacionais baixos, ao recomputar uma nova mediana para o agrupamento C_i, somente estações candidatas $\zeta_k \mid \zeta_k \in LP_i$ são consideradas, sendo LP_i a Lista de Proximidade da atual mediana ζ_i do agrupamento C_i. O mesmo vale para o agrupamento C_j.

Este processo continua até que todos os pares de agrupamentos sejam avaliados para todos os valores possíveis de μ e μ', sem repetições. Se nenhuma melhoria na função objetivo for alcançada, então é provável que um mínimo local tenha sido atingido. Para escapar deste possível mínimo local, o Conjunto de Listas de Proximidade (CLP) é recalculado para um novo κ, adicionando-lhe um $\Delta\kappa$. À medida que os tamanhos das listas de proximidade crescem, novas vizinhanças podem ser alcançadas.

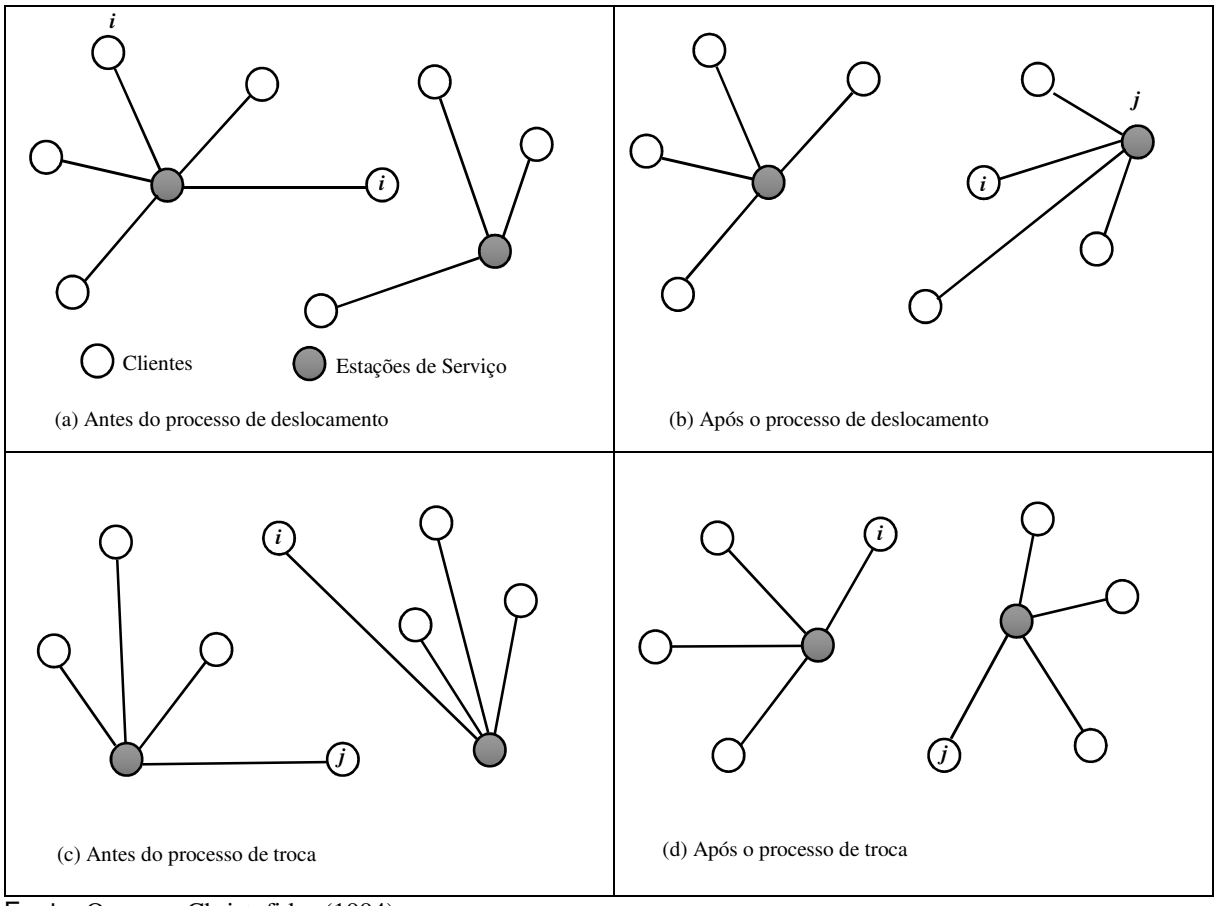
Para evitar que o tamanho de cada Conjunto de Listas de Proximidade cresça muito, invalidando assim o seu principal objetivo de reduzir o número total de operações de deslocamento e troca, só é permitido que κ cresça até que o tamanho médio de todas as listas de proximidade que compõem o CLP, atinja 20% do número de estações candidatas, *J*. Quando este limite é atingido e se ainda não houver nenhuma melhoria na função objetivo, uma mediana ζi da solução incumbente *S* é trocada aleatoriamente por outro vértice $\zeta j \mid \zeta j \notin S, \zeta j \in LPi,$ sendo *LPi* a Lista de Proximidade da mediana ζi. Ambas as medidas combinadas devem permitir que novas vizinhanças sejam alcançadas nas iterações subsequentes.

O processo termina quando um número específico de iterações sem melhorias é alcançado. O número de iterações é proporcional ao *NumHit* da molécula, que armazena o número total de colisões que uma molécula sofreu até um dado momento. O limite exato de iterações, para uma determinada execução do mecanismo *λ-interchange*, é determinado pelo produto de *NumHit* por um número mínimo de iterações (*Minλ-interchangeIterations*), que normalmente é um inteiro de valor pequeno, como 1 ou 2. Isso permite que as primeiras reações ineficazes ocorram muito rapidamente, descartando moléculas que não carregam soluções promissoras. Estas moléculas em poucas iterações, se tornam alvos para decomposições e sínteses, devido à perda acentuada de energia cinética, que ocorre sempre que não atinjam melhorias rápido o suficiente. Por outro lado, moléculas que carregam soluções melhores tendem a sobreviver por mais tempo, pois a cada nova colisão elas recebem um limite de iterações mais alto.

De acordo com Osman e Christofides (1994), uma solução S é *λ*-ótima *(λ-opt,* do inglês *λ-optimal*) se, e somente se, para qualquer par de agrupamentos $Ci, Cj, \in S$, não houver nenhuma melhoria que possa ser feita por qualquer operação do mecanismo *λ-interchange*. Os autores também afirmam que, para produzir um algoritmo de descida *λ-opt* eficiente, é aconselhável produzir primeiro uma solução que seja *1-opt*. Na presente implementação do *λ-interchange* segue-se esta recomendação, obtendo-se sempre uma solução *1-opt* antes de tentar melhorá-la para uma solução *2-opt*. Devido aos altos custos computacionais de tal mecanismo, *λ* foi

limitado a um valor máximo de dois. Todas as colisões ineficazes com a parede obtêm uma solução *1-opt*, enquanto que as colisões ineficazes intermoleculares obtêm uma solução *2-opt*, se necessário. A relação aproximada de obtenção de soluções *1-opt* em relação a *2-opt* é 10, e é controlada configurando-se o parâmetro *Molecoll*, do CRO, para 0,1. O pseudocódigo parcial do mecanismo *λ-interchange* é mostrado na Figura 18.

Figura 17 - O mecanismo *λ –interchange*, para *λ* = 1 (*1-interchange*)



Fonte: Osman e Christofides (1994)

Figura 18 – O mecanismo *λ-interchange* para o CRO

(continua)

```
class LambdaInterchangeAlg
properties:
   enumeration Operations { ShiftAtoB, ShiftBtoA, InterchangeAB } // To track all λ-interchange operations
Methods:
// Arguments: λ = target λ-opt  /  maxIt = Max iterations  / S = Current solution to be improved
Solve ( λ, maxIt, S : Solution ) : Solution
{
   //Compute new Proximity List for initial κ
   κ ← κ0;   CLP  ← new CLP (κ);

   // Save best solution
   S* ← S

  // Perform maxIt iterations
   for (it ← 0, it < maxIt, it++)
   {
      // To avoid redundant operations such as (0,2) and (0,2)
      previousμ ← 0;   previousμ´ ← 0;

     // Loop from 1 to λ
     for (μ ← 1, μ ≤ λ, μ++)
      {
         // Loop from 1 to λ
         for (μ´ ← 1, μ´ ≤ λ, μ´++)
         {
            for each op in Operations // Loops through all operations
            {
               // Avoid redundant interchange/shift operations
               if ( ( op = Operations.InterchangeAB) OR  ( ( op = Operations.ShiftAtoB)
           AND (μ > previousμ)) OR  ( ( op = Operations.ShiftBtoA) AND (μ´ > previousμ´) ))
               {
                  // Update previousμ and previousμ´
                  if (op = Operations.ShiftAtoB )   previousμ ← μ;
                 else  if (op = Operations.ShiftBtoA )  previousμ´ ← μ´;

                 // Build list of stations that are part of the current solution (medians)
                 StationListA ← new List of stations from S;

                 // Loops through all stations in StationListA
                 while (StationListA.Count > 0)
                 {
                    // Randomly pick a station from StationListA
                    stationA ← random station from StationListA;

                    // Select μ customers from stationA, without repetition
                    for each μcustomersA in (distinct μ customers from stationA)
                    {
                       // Build list of stations that belong to the proximity list of stationA
                       StationListB ← new List of stations from LP[stationA];

                       // Loops through all stations in StationListB
                       while (StationListB.Count > 0)
                       {
                          // Randomly pick a station from StationListB
                          stationB ← random station from StationListB

                           // To avoid processing redundant combinations of stations, i.e., (stA,stB) and (stB,stA)
                          if (  (stationA < stationB) OR
                              ((stationA > stationB) AND (stationA NOT IN LP[stationB]) ))
                          {
```

Figura 18– O mecanismo λ-interchange para o CRO

(conclusão)

```
                        // Select μ´ customers from stationB, without repetition
                        for each μ´customersB in (distinct μ´customers from stationB)
                        {
                           // Generate neighborhood solution S' by carrying out the operation specified in op
                           // against μcustomersA and μ´customersB
                           // e.g. Interchanging μ customers from stationA with μ´ customers from stationB
                           S' ← MoveEdges (S, stationA, μcustomersA, stationB, μ´customersB);

                           // If a better solution is found
                           if (S´.PE < S.PE)
                           {
                              // Update S and recompute medians for all affected clusters
                              S ← S´;
                              S ← SolveOneMedian (S, stationA);
                              S ← SolveOneMedian (S, stationB);
                           }
                        } // for each μ´customersB in (distinct μ´customers from station)
                     } // if ((stationA < stationB) OR...

                     // Remove stationB from StationListB
                     stationB.Remove (stationB);

                  } // while (StationListB.Count > 0)
               } // for each μcustomersA in (distinct μ customersA from stationA)

               // Remove stationA from StationListA
               StationListA.Remove (stationA);

            } while (StationsA.Count > 0)
         } // if ((op = Operations.InterchangeAB)...
      } // for each op in Operations
   } // for (μ´ ← 1, μ´ ≤ λ, μ´++)
} // for (μ ← 1, μ ≤ λ, μ++)

// Update S* if a better solution is found
if (S.PE < S*.PE)
   S* ← S;
else
{
   // Additionally, increase size of CLPk by Δk
   // if average size of all proximity lists in the set is below 20% of J
   if ( Average (LP[1].Count, LP[2].Count,...,LP[j].Count) < (J / 5) )
   {
      κ ← κ + Δk;
      LP  ← new CLP (κ);
   }
   else
   // if solution not improved, exchange one median with another that is not part of the current solution.
   S ← ExchangeOneMedian (S, random station j from J | j ∉ S );
}
} // for (it ← 0, it < maxIt, it++)
// Return best solution S*
return S*;
} // Solve
End class
```

Fonte: Elaborado pelo autor (2017)

3.6.6 O operador Half-Total change (HTC)

É o operador utilizado para efetuar a decomposição de uma molécula em duas outras moléculas, conforme descrito no item 3.6.8.3. Como o seu nome sugere, uma nova solução é produzida a partir de uma já existente, mantendo-se metade dos valores (medianas) existentes e atribuindo-se novos valores à metade restante.

Suponha que se tente produzir duas novas soluções $\omega'_1 = [\omega'_1(i), 1 \leq i \leq p]$ e $\omega'_2 = [\omega'_2(i), 1 \leq i \leq p]$ *a partir de* $\omega = [\omega(i), 1 \leq i \leq p]$. Para obter-se ω'_1, primeiro copia-se ω para ω'_1 e, em seguida, escolhe-se aleatoriamente $\lfloor N/2 \rfloor$ elementos no vetor ω'_1, onde $\lfloor . \rfloor$ retorna o maior inteiro igual ou menor que o argumento. Para cada um destes elementos escolhidos, por exemplo $\omega'_1(i)$, atribui-se um novo valor de mediana, respeitando-se as restrições do problema. Como os elementos são escolhidos aleatoriamente, ω'_1 e ω'_2 são bastante diferentes entre si, e também de ω'. Na presente implementação, o operador *Half-total change* assegura que os elementos de ω' estejam presentes em ω'_1 ou ω'_2, mas não em ambos. Isto garante que todas as estações pertencentes a ω' estejam presentes, seja em ω'_1 ou ω'_2.

O operador HTC tem como entrada a *MinStruct* da molécula, que armazena a melhor solução atingida pela mesma, retornando duas soluções de saída ω'_1 e ω'_2. Inicialmente o processo descrito acima é repetido 100 vezes, para cada solução de saída gerada. Então, as melhores soluções são escolhidas para ser tornarem ω'_1 e ω'_2. Antes de serem retornadas ao algoritmo principal, ambas soluções são melhoradas, passando pelo processo descrito nos itens 3.6.2c a 3.6.2f.

3.6.7 O operador Distance Preserving Crossover (DPX)

É o operador usado para efetuar a síntese de duas moléculas em uma única molécula. O operador DPX foi utilizado por Merz e Freisleben (1997) para resolução do problema de assinalamento quadrático, através de Algoritmo Genético, se mostrando bem adaptado ao CRO, uma vez que se baseia unicamente na noção de distância entre soluções. Sejam ω_1 e ω_2 soluções válidas para um dado problema. A distância *T*, entre soluções é definida como:

$$T(\omega_1, \omega_2) = |\{i \in \{1, \dots, n\} | \omega_1(i) \neq \omega_2(i)\}| \quad (15)$$

Tal como definido em (15), *T* representa quantidade de estações presentes em ω_1 que não se se encontram em ω_2. O DPX tem como objetivo produzir um descendente que possua a mesma distância de cada um de seus pais, sendo esta distância igual à distância entre os próprios pais.

Sejam dois pais A e B, contendo soluções viáveis. Primeiro, todas as atribuições de estações contidas em ambos os pais são copiadas para o descendente C. As posições restantes são, então, aleatoriamente preenchidas com estações ainda não atribuídas, assegurando-se que nenhuma atribuição que possa ser encontrada em apenas um dos pais seja inserida no descendente. Desta forma, obtém-se um descendente *C*, para qual a condição $T(C,A) = T(C,B) = T(A,B)$ se mantém. Tal cruzamento é altamente disruptivo, forçando as subsequentes buscas locais a explorarem uma região diferente do espaço de solução, onde melhores soluções poderão ser encontradas. Se uma solução viável não puder ser encontrada, a operação falha, sendo a síntese rejeitada.

O operador DPX toma como entrada a *MinStruct* de duas moléculas, ω_1 e ω_2, que contêm as melhores soluções atingidas pelas mesmas, retornando uma única molécula, ω'. Tal processo é repetido 100 vezes. Então, a molécula com o *PE*, que no CRO corresponde ao valor da função objetivo, mais baixo é selecionada e retornada como ω'. Antes de ser retornada ao algoritmo principal, a nova solução, ω', é melhorada, passando pelo processo descrito nos itens 3.6.2c a 3.6.2f.

3.6.8 Reações Elementares

Existem quatro tipos de reações elementares que podem ocorrer a cada iteração do CRO. Elas são empregadas para manipular soluções (explorar o espaço de solução) e redistribuir energia entre as moléculas e o *buffer* de energia. Operadores são usados para modificar as soluções ou gerar novas soluções a partir das soluções correntes. No entanto, o CRO sempre garante a conservação da energia quando novas soluções são geradas através dos operadores.

Para as colisões ineficazes com a parede e intermoleculares, foi utilizado um operador baseado no algoritmo denominado "*λ-interchange mechanism*", proposto por Osman e Christofides (1994), conforme descrito no item 3.6.5. Nas sínteses, o operador DPX foi utilizado, conforme descrito no item 3.6.7. Finalmente, nas decomposições foi empregado o operador *Half-total change*, descrito no item 3.6.6.

Se uma solução produzida por qualquer um destes operadores causar alguma violação das restrições do modelo sendo resolvido (CPMP ou FCLP Baouche) a mesma é rejeitada. Adicionalmente, para que uma solução seja aceita, a mesma deve passar pelos mecanismos de aceitação, baseados em energia, próprios do CRO, que estão descritos sucintamente nesta dissertação. Maiores informações sobre o funcionamento interno do CRO podem ser encontrados em Lam e Li (2012).

3.6.8.1 Colisão ineficaz com a parede

Ocorre quando uma molécula colide com a parede de um recipiente e então é rebatida, sem se partir, permanecendo uma única molécula. Neste tipo de colisão, a solução existente, ω, é perturbada e se torna uma solução ω', i.e., $\omega \rightarrow \omega'$.

Isto é feito gerando uma solução ω' que esteja na vizinhança de ω, através de um operador do tipo *λ-interchange*, proposto por Osman e Christofides (1994) e descrito no item 3.6.5.

Seja *N(.)* um operador de busca em vizinhança do tipo *λ-interchange*. Tem-se, então, que $\omega' = N(\omega)$ e $PE\omega' = f(\omega')$. Nesse tipo de reação, tipicamente, ocorrerá uma perda de energia potencial, ou seja, $PE\omega'$ será menor que $PE\omega$, indicando que uma solução melhor foi obtida. Caso isto não ocorra e $PE\omega'$ seja maior que $PE\omega$, ainda assim a solução pior poderá ser aceita desde que $PE\omega + KE\omega \geq PE\omega'$. No entanto, toda vez que uma reação elementar ocorre, uma certa quantidade de energia cinética (*KE*) é transferida para o *buffer* de energia, diminuindo as chances de que soluções piores sejam aceitas à medida que as iterações ocorrem.

A quantidade de energia cinética da molécula obtida a partir da reação ineficaz é indiretamente controlada pelo parâmetro *KElossRate,* que é um valor entre 0 e 1, inclusive, e determina a quantidade mínima de energia cinética que será aproveitada da solução original (ω). Por exemplo, se à *KElossRate* for atribuído o valor 0,8, um mínimo de 80% da energia cinética que a molécula possuía antes da colisão será transferida para a molécula resultante, após a colisão. A diferença de energia cinética será, então, transferida para o *buffer* central de energia. O pseudocódigo para a colisão ineficaz com a parede está mostrado na Figura 35.

3.6.8.2 Colisão ineficaz intermolecular

Ocorre quando duas moléculas colidem entre si e, em seguida, se afastam. O número de moléculas permanece inalterado, i.e., $\omega_1 + \omega_2 \rightarrow \omega'_1 + \omega'_2$. Esta reação é muito similar à colisão ineficaz com a parede. Desta forma o mesmo operador baseado no mecanismo *λ-interchange*, descrito no item 3.6.5, é utilizado para gerar novas vizinhanças. Seja *N(.)* um operador to tipo *λ-interchange*. Desta forma, ω'_1 e ω'_2 são obtidos por $\omega'_1 = N(\omega_1)$ e $\omega'_2 = N(\omega_2)$. O gerenciamento de energia é similar à da colisão ineficaz com a parede, mas não envolve o *buffer* de energia.

3.6.8.3 Decomposição

Ocorre quando uma molécula (ω) colide com uma parede e, em seguida, quebra-se em duas partes, produzindo ω'_1 *e* ω'_2, ou seja: $\omega \rightarrow \omega'_1 + \omega'_2$

O objetivo da decomposição é permitir que o sistema explore outras regiões do espaço de soluções, após ter efetuado considerável busca local através de colisões ineficazes. No presente trabalho, utilizou-se o algoritmo *Half-total change*, descrito no item 3.6.6, para gerar novas soluções. Uma vez que são criadas mais soluções, a soma total de *PE* e *KE* da molécula original pode não ser suficiente para que a transformação seja aceita. Em outras palavras, pode ocorrer que $PE\omega + KE\omega < PE\omega'_1 + PE\omega'_2$. Como a conservação de energia não é satisfeita nestas condições, esta decomposição deve ser abortada. Para aumentar a chance de ter-se uma decomposição concluída, uma pequena porção de energia do *buffer* é retirada para apoiar a mudança. O pseudocódigo da decomposição está mostrado na Figura 36.

3.6.8.4 Síntese

A síntese é o oposto da decomposição. Uma síntese acontece quando duas moléculas colidem uma contra a outra e se fundem, isto é: $\omega_1 + \omega_2 \rightarrow \omega'$

Nesta reação, permite-se uma mudança muito maior para ω', em relação a ω_1 *e* ω_2, bem como um aumento considerável da energia cinética da molécula resultante. Assim a mesma possui uma maior "capacidade" de explorar seu espaço de soluções, devido a sua energia cinética superior. A ideia por trás da síntese é a diversificação de soluções. No presente trabalho, um operador denominado *Distance Preserving Crossover* (DPX) comumente empregado em Algoritmos Genéticos (GA) foi utilizado. O operador DPX é descrito no item 3.6.7. O pseudocódigo da síntese está mostrado na Figura 38.

3.6.9 Conservação da Energia

Uma das premissas do CRO é a da conservação da energia, o que significa que a energia não pode ser criada e nem destruída. O sistema, em sua totalidade, compreende a energia de todas as moléculas existentes, somada à energia contida no recipiente, que está associada ao buffer. A quantidade total inicial de energia de todo o sistema é determinada pelos valores da função objetivo, isto é, a energia potencial (*PE*) da população inicial de moléculas, cujo tamanho é determinado por *PopSize*, a energia cinética (*KE*) inicial (*InitialKe*) atribuído às moléculas e

valor inicial de energia do buffer. Em todos os experimentos o valor inicial para o *buffer* de energia foi zero.

3.6.10 Inicialização

Inicialmente, atribuem-se valores iniciais às variáveis escalares correspondentes aos parâmetros operacionais padrão do CRO, ou seja, ***PopSize, KELossRate, MoleColl, buffer, InitialKE, α*** e ***β***. Adicionalmente, alguns outros parâmetros foram introduzidos na presente implementação para o CPMP e modelo de Baouche e colaboradores (2014):

- ***λ* :** controla maior solução *λ-opt* a ser obtida pelo mecanismo *λ-interchange*. Por exemplo, se $\lambda = 2$, somente soluções *1-opt* e *2-opt* serão produzidas pelo mecanismo.
- ***Minλ-interchangeIterations***: é o valor inicial de iterações sem melhorias a serem executadas pelo mecanismo *λ-interchange*. O valor atual, ou corrente, para uma dada molécula é dado por *Minλ-interchangeIterations* * (*NumHit + 1)*.
- ***MinMol***: é o número mínimo de moléculas da população. Se a população atinge este valor limite nenhuma síntese poderá ocorrer.
- ***MaxMol***: é o número máximo de moléculas da população. Se a população atinge este valor limite nenhuma decomposição poderá ocorrer.
- ***MaxIterations***: é o número máximo de iterações a serem realizadas pelo algoritmo principal do CRO.
- ***MaxIterationsWithoutImprovement***: é o número máximo de iterações a serem realizadas pelo algoritmo principal do CRO, sem que existam melhorias na função objetivo.
- ***κ₀* :** é o valor inicial para o fator de expansão de capacidade (*κ)*, do Conjunto de Listas de Proximidade (CLP). É usado pelo mecanismo *λ-interchange* para indiretamente influenciar o tamanho das listas de proximidade do CLP, conforme explicado nos itens 3.6.4 e 3.6.5.
- ***Δκ*** : incremento do fator de expansão de capacidade (*κ),* usado no Conjunto de Listas de Proximidade (CLP) pelo mecanismo *λ-interchange*, como explicado no item 3.6.5.

Em seguida, um algoritmo construtivo, explicado no item 3.6.2, é invocado para criar uma população inicial de moléculas, de tamanho *PopSize*.

3.6.11 Iterações e finalização

Durante a fase iterativa do CRO, uma molécula pode atingir uma parede do recipiente ou colidir com uma outra molécula. Isto é decidido gerando-se um número aleatório *b* entre [0, 1]. Se $b > MoleColl$ ou se o sistema tiver apenas uma molécula, uma colisão unimolecular ocorrerá. Caso contrário, ocorrerá uma colisão intermolecular. Para uma colisão unimolecular, uma molécula da população é selecionada aleatoriamente e, então, decide-se se ocorrerá com ela uma colisão ineficaz com a parede ou uma decomposição, de acordo com o critério de decomposição definido.

No presente trabalho, o critério de decomposição foi definido como:

$$NumHit - MinHit > \alpha \tag{16}$$

De forma análoga, para uma colisão intermolecular, selecionam-se aleatoriamente duas moléculas da população e, então, determina-se se haverá uma colisão ineficaz intermolecular ou uma síntese verificando-se o critério de síntese para as moléculas escolhidas. A síntese ocorrerá se todas as moléculas atenderem o seguinte critério:

$$KE \leq \beta \tag{17}$$

As desigualdades (16) e (17) controlam o grau de diversificação através dos parâmetros α e β. Valores adequados de α e β promovem um equilíbrio entre diversificação e intensificação.

Após a ocorrência de uma reação elementar, deve-se verificar se a condição de conservação de energia é obedecida. Caso isso não haja ocorrido a mudança deve ser descartada. A seguir, é verificada se qualquer uma das soluções produzidas naquela operação possui um valor de função objetivo mais baixo do que a melhor solução obtida até o momento. Se assim o for, registra-se tal solução como a melhor até o momento.

Se nenhum critério de parada for atingido, inicia-se uma nova iteração, após armazenar-se a melhor solução dentre as melhores soluções de toda a população, ou seja, após armazenar-se o *MinStruct* com o menor *MinPE* entre todas as moléculas.

Se nenhum critério de parada for atingido, inicia-se uma nova iteração, após armazenar-se a melhor solução dentre as melhores soluções de toda a população, ou seja, após armazenar-se o

MinStruct com o menor *MinPE* entre todas as moléculas. Caso contrário, termina-se o ciclo iterativo, retornando-se a melhor solução encontrada.

O número de iterações é controlado pelo parâmetros *MaxIterations* e *MaxIterationsWithoutImprovement*, descritos no item 3.6.10. O algoritmo principal do CRO, modificado para o CPMP e FCLP Baouche, é mostrado na Figura 19 .

Os pseudocódigos para ambas as reações ineficazes, síntese e decomposição não foram alterados em relação aos do tutorial no qual são baseados, e são mostrados nas Figura 35, Figura 36Figura 37 e Figura 38. Maiores informações sobre o seu funcionamento e implementação podem ser obtidas em Lam e LI (2012).

Figura 19 – Algoritmo principal do CRO para os modelos CPMP e FCLP Baouche

```
class CROAlg
Properties:
   PopSize, KELossRate, MoleColl,buffer,InitialKE,α,β,
   MinMoll, MaxMoll, MaxIterations, MaxIterationsWithoutImprovement
Methods:
Solve ( PopSize, KELossRate, MoleColl,buffer,InitialKE,α,β,
      MinMoll, MaxMoll, MaxIterations, MaxIterationsWithoutImprovement ) : Solution
{
   // Initialization: Create molecule populaton of size PopSize
   CreateMolecules (PopSize, InitialKE);

   // Iterations
   it ← 1;
   itSameObj ← 1;
   bestObj = ∞;
   bestSolution = null;
   while (it ≤ MaxIterations) AND (itSameObj ≤ MaxIterationsWithoutImprovement)
   {
      Generate b in [0, 1];
      if (b > MoleColl) OR (PopSize = 1)
      {
         Randomly select one molecule Mω from population;

         if (Mω.NumHit - Mω.MinHit) > α // Decomposition criterion met
            if (PopSize < MaxMoll) // PopSize cannot exceed MaxMoll
               Decomposition (Mω); // Trigger decomposition
         else
            OnwallIneffectiveCollision (Mω); // Trigger On-wall Ineffective Collision
      }
      else
      {
         Randomly select two molecules Mω1 and Mω2;

         if (Mω1.KE < β) AND (Mω2.KE < β) // Synthesis criterion met
         {
            if (PopSize > MinMoll) // PopSize cannot b eless than MinMoll
               Synthesis (Mω1,Mω2);  // Trigger Synthesis
            else
               IntermolecularIneffectiveCollision (Mω1,Mω2); // Trigger Intermol. Ineffective Collision
         }
      }
      // Check for any new minimum solution
      if (best MinPE in population < bestObj)
      {
         bestObj ← MinPE in population;
         bestSolution ← Solution with MinPE in population;
      }
      else
         itSameObj ← itSameObj + 1;
   } // main loop

   // The final stage: Output the best solution found and its objective function value
   return bestSolution
} // Solve
End class
```

Fonte: Elaborado pelo autor (2017)

3.7 CRIAR UMA BASE DE DADOS, COMPOSTA DE AGRUPAMENTOS DE EVS E ESTAÇÕES DE CARREGAMENTO CANDIDATAS, A PARTIR DE DADOS OBTIDOS DE SERVIÇOS DE MAPEAMENTO, BUSCA E NAVEGAÇÃO, DISPONÍVEIS NA INTERNET

Conforme descrito na Etapa 3.1, uma base de dados contendo locais candidatos a receberem estações de carregamento de veículos elétricos na Grande Vitória, ES foi criada, utilizando-se o serviço online de mapeamento *Google Maps API* (GOOGLE INC., 2018).

Tal base, denominada *vix,* foi montada a partir de 272 pontos de interesse, obtidos através de consultas ao *Google Maps API*. A mesma é composta de 5 instâncias, com tamanhos entre 50 e 250 vértices (estações candidatas/centros de consumo), a partir de pontos selecionados randomicamente dentre os 272 disponíveis. O processo de criação das instâncias é descrito a seguir:

3.7.1 Seleção da localização das estações candidatas e centros de consumo

Foram selecionados a partir de 272 pontos de interesse através da *Google Places API Web Service* (GOOGLE INC, 2018), que é um serviço que retorna informações sobre locais — definidos pela API como estabelecimentos, localizações geográficas ou pontos de interesse proeminentes — usando requisições HTTP.

Num primeiro momento, definiram-se quais categorias de estabelecimentos comerciais e outros locais poderiam se tornar candidatos a receber estações de carregamento. Dentre os diversos tipos de categorias consideradas foram selecionados postos de gasolina, hospitais, estacionamentos comerciais e shopping centers. Além disso, decidiu-se incluir o Aeroporto de Vitória. Tais categorias correspondem a locais típicos onde estações comerciais de carregamento de EVs têm sido instaladas nos Estados Unidos (MCCAULEY, 2017) e alguns países da Europa.

Uma vez que as categorias de locais foram definidas, pontos centrais a cada uma das cidades que compõem a Grande Vitória foram determinados, após consulta a uma base gratuita de informações geográficas (DBCITY, 2017). As coordenadas obtidas, em termos de latitude e longitude, estão mostradas na Tabela 3.

Através de um programa desenvolvido em linguagem de programação *Transact-SQL* (*T-SQL*) para execução em um servidor de bancos de dados *Microsoft SQL Server 2016*, requisições

HTTP foram submetidas ao serviço *Google Places API Web Service*, procurando locais candidatos, num raio de até 50Km das coordenadas correspondentes às cidades da Grande Vitória. Os resultados das consultas, em formato XML, foram salvos em uma tabela de banco de dados para filtragem posterior. O código parcial do programa de busca está mostrado na Figura 20.

Com os locais retornados pelas consultas armazenados na forma de linhas de uma tabela, efetuou-se um pós-processamento de filtragem com o intuito de remover, de forma automática, locais não estivessem situados dentro dos municípios da Grande Vitória, além de locais retornados pela *Google Places API* que não faziam sentido, do ponto de vista de localização de estações de carregamento de EVs, como teatros, cinemas, academias de ginástica, lojas de roupas, etc. O código parcial usado na filtragem está mostrado na Figura 21.

Por fim, uma verificação manual, buscando eliminar linhas referenciando locais inadequados, foi posteriormente efetuada, reduzindo o tamanho inicial da tabela de 547 para 272 locais. Uma visão parcial da tabela de locais candidatos é mostrada na Figura 22.

Tabela 3 – Latitude e Longitude das Cidades da Grande Vitória, ES

Cidade	**Latitude**	**Longitude**
Vitória	-20.3222	-40.3381
Vila Velha	-20.3305	-40.2922
Cariacica	-20.2655	-40.4203
Serra	-20.1294	-40.308
Guarapari	-20.6511	-40.5067
Fundão	-19.9352	-40.4128
Viana	-20.3893	-40.4948

Fonte: DBCity (2017)

Figura 20 – Código *T-SQL* para busca na *Google Places API*

```
Use GoogleMapsDB
...
-- Set Google API URL and Developer Key
SET @Key = 'AIzaSyBCbfX3Vp-Y1Vm8LkazTI8RigHZ1-jgqho'
SET @UrlBase = 'https://maps.googleapis.com/maps/api/place/nearbysearch/xml?'

-- Populate table with URLs to be processed
-- Vitoria
Set @Coord = '-20.3222,-40.3381'
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=gas_station&keyword=posto&language=pt-BR&key=') -- Gas Stations
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=airport&keyword=''aeroporto de vitória''&language=pt-BR&key=') -- Airports
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=hospital&keyword=''hospital''&language=pt-BR&key=') -- Hospitals
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=parking&keyword=''estacionamento''&language=pt-BR&key=') -- Parking
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=shopping_mall&keyword=shopping&language=pt-BR&key=') -- Shopping Centers
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=shopping_mall&keyword=mall&language=pt-BR&key=') -- Shopping Centers
-- Vila Velha
Set @Coord = '-20.3305,-40.2922'
insert into #URLs (PlaceURL) VALUES ('location='+@Coord+'&radius=50000&type=gas_station&keyword=posto&language=pt-BR&key=') -- Gas Stations
...
-- Cariacica
Set @Coord = '-20.2655,-40.4203'
...
-- Loop through URLs
Declare @PlaceURL varchar(1024)
declare URLsCursor Cursor FOR Select PlaceURL from #URLs order by ID
OPEN URLsCursor
FETCH NEXT FROM URLsCursor INTO @PlaceURL
WHILE @@FETCH_STATUS = 0
BEGIN
    ...
    -- Set URL and start looking for places
    SET @UrlOrig = @UrlBase + @PlaceURL + @Key
    ...
    -- Create OLE Automation object to send HTTP requests via SQL Server
    EXEC @err = sys.sp_OACreate @progid = 'MSXML2.ServerXMLHTTP', @objecttoken = @obj OUT, @context = 1
    ...
    -- Send HTTP request and save response on table
    insert into #singleXMLresponse exec @err =sp_OAGetProperty @obj, 'ResponseText'
    ...
    -- Cast response to XML
    SET @resposta = (Select TOP 1 res from #singleXMLresponse)
    ...
```

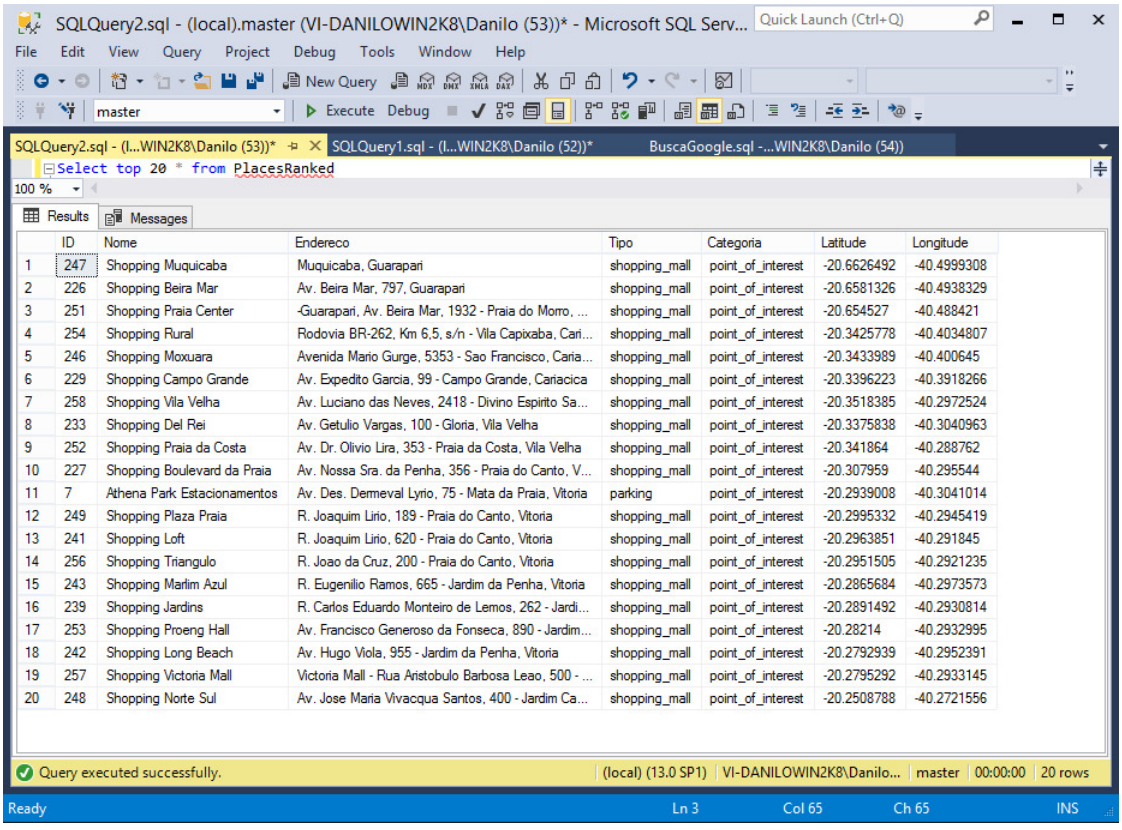
Fonte: Elaborado pelo autor (2017)

Figura 21 – Código *T-SQL* para filtragem de locais inapropriados ou inválidos

```
-- Apply filter to places table to remove invalid records
Select IDENTITY(int,1,1) as ID, Nome,Endereco,Tipo,Categoria,[Latitude], [Longitude]
Into PlacesRanked
from #Places
where (Nome <> '') and (Nome is not null)
    and (  (Endereco like '%Vitoria%') or (Endereco like '%Vila Velha%') or
    (Endereco like '%Cariacica%') or (Endereco like '%Serra%') or
        (Endereco like '%Fundao%') or (Endereco like '%Guarapari%') or
    (Endereco like '%Viana%') )
    and (Categoria not in
    ('store','gym','travel_agency','clothing_store','movie_theater'))
    and (Tipo in ('shopping_mall','hospital', 'parking','airport','gas_station'))
    and ( (Nome like '%Hospital%') OR (Nome like '%Posto%')
    or (Nome like '%Estacionamento%') OR (Nome like '%Aeroporto%')
    or (Nome like '%Shopping%') )
    and ( (Nome not like '%Lojas%') )
order by Nome
```

Fonte: Elaborado pelo autor (2017)

Figura 22 – Tabela de locais candidatos obtida pela *Google Places API*



	ID	Nome	Endereco	Tipo	Categoria	Latitude	Longitude
1	247	Shopping Muquicaba	Muquicaba, Guarapari	shopping_mall	point_of_interest	-20.6626492	-40.4999308
2	226	Shopping Beira Mar	Av. Beira Mar, 797, Guarapari	shopping_mall	point_of_interest	-20.6581326	-40.4938329
3	251	Shopping Praia Center	-Guarapari, Av. Beira Mar, 1932 - Praia do Morro, ...	shopping_mall	point_of_interest	-20.654527	-40.488421
4	254	Shopping Rural	Rodovia BR-262, Km 6,5, s/n - Vila Capixaba, Cari...	shopping_mall	point_of_interest	-20.3425778	-40.4034807
5	246	Shopping Moxuara	Avenida Mario Gurge, 5353 - Sao Francisco, Caria...	shopping_mall	point_of_interest	-20.3433989	-40.400645
6	229	Shopping Campo Grande	Av. Expedito Garcia, 99 - Campo Grande, Cariacica	shopping_mall	point_of_interest	-20.3396223	-40.3918266
7	258	Shopping Vila Velha	Av. Luciano das Neves, 2418 - Divino Espirito Sa...	shopping_mall	point_of_interest	-20.3518385	-40.2972524
8	233	Shopping Del Rei	Av. Getulio Vargas, 100 - Gloria, Vila Velha	shopping_mall	point_of_interest	-20.3375838	-40.3040963
9	252	Shopping Praia da Costa	Av. Dr. Olivio Lira, 353 - Praia da Costa, Vila Velha	shopping_mall	point_of_interest	-20.341864	-40.288762
10	227	Shopping Boulevard da Praia	Av. Nossa Sra. da Penha, 356 - Praia do Canto, V...	shopping_mall	point_of_interest	-20.307959	-40.295544
11	7	Athena Park Estacionamentos	Av. Des. Demeval Lyrio, 75 - Mata da Praia, Vitoria	parking	point_of_interest	-20.2939008	-40.3041014
12	249	Shopping Plaza Praia	R. Joaquim Lirio, 189 - Praia do Canto, Vitoria	shopping_mall	point_of_interest	-20.2995332	-40.2945419
13	241	Shopping Loft	R. Joaquim Lirio, 620 - Praia do Canto, Vitoria	shopping_mall	point_of_interest	-20.2963851	-40.291845
14	256	Shopping Triangulo	R. Joao da Cruz, 200 - Praia do Canto, Vitoria	shopping_mall	point_of_interest	-20.2951505	-40.2921235
15	243	Shopping Marlim Azul	R. Eugenilio Ramos, 665 - Jardim da Penha, Vitoria	shopping_mall	point_of_interest	-20.2865684	-40.2973573
16	239	Shopping Jardins	R. Carlos Eduardo Monteiro de Lemos, 262 - Jardi...	shopping_mall	point_of_interest	-20.2891492	-40.2930814
17	253	Shopping Proeng Hall	Av. Francisco Generoso da Fonseca, 890 - Jardim...	shopping_mall	point_of_interest	-20.28214	-40.2932995
18	242	Shopping Long Beach	Av. Hugo Viola, 955 - Jardim da Penha, Vitoria	shopping_mall	point_of_interest	-20.2792939	-40.2952391
19	257	Shopping Victoria Mall	Victoria Mall - Rua Aristobulo Barbosa Leao, 500 - ...	shopping_mall	point_of_interest	-20.2795292	-40.2933145
20	248	Shopping Norte Sul	Av. Jose Maria Vivacqua Santos, 400 - Jardim Ca...	shopping_mall	point_of_interest	-20.2508788	-40.2721556

Fonte: Elaborado pelo autor (2017)

3.7.2 Determinação dos padrões de deslocamento veicular

Conforme estabelecido no item dos objetivos específicos, utilizaram-se rotas reais de deslocamento veicular entre os centros de consumo e as estações de carregamento de EVs. Tais informações foram obtidas através do serviço *Google Maps Distance Matrix API*, que que fornece a distância e o tempo de percurso para uma matriz de origens e destinos (GOOGLE INC., 2018).

Desta forma, um programa, em linguagem de programação *Transact-SQL* (*T-SQL*), foi desenvolvido para submeter requisições HTTP ao *Google Maps Distance Matrix API*, com o objetivo de determinar as distâncias de percurso, de automóvel, entre cada local armazenado na tabela de locais, gerada no item 3.7.1. Assim, para cada local da tabela, foram calculadas as distâncias de percurso até todos os outros locais da mesma. Os resultados foram armazenados em outra tabela contendo uma matriz de distâncias de percurso.

É importante mencionar que como se tratam de distâncias obtidas de rotas reais, a distância entre dois pontos *a* e *b*, pode não ser a mesma entre *b e a*, já que os percursos de ida e vinda quase sempre são diferentes, mesmo que ligeiramente, na maioria dos casos. O código parcial do programa em T-SQL que gera a matriz de distâncias está mostrado na Figura 23.

3.7.3 Geração das instâncias

A partir dos locais candidatos, obtidos no item 3.7.1, e das distâncias de percurso, obtidas no item 3.7.2, cinco instâncias foram criadas com locais selecionados randomicamente dentre os 272 disponíveis. Cada local selecionado serviu tanto como candidato a se tornar uma estação de carregamento de EVs como um centro de consumo. A razão de usar-se os mesmos locais para ambas as finalidades se deve ao fato dos mesmos se situarem em áreas de maior concentração e circulação de veículos (como postos de gasolina, shopping centers, estacionamentos comerciais e outros) onde deveria haver tanto uma maior necessidade de carregamento por parte dos EVs como uma disponibilidade maior de estações. A Figura 26 exibe, num mapa da Grande Vitória, os locais onde as estações de carregamento foram localizadas, para uma das instâncias criadas (*vix100*). Os círculos representam os centros de consumo, onde estão concentradas as demandas de carregamento, e os marcadores numerados em forma de gota, as estações de carregamento selecionadas. Como pode-se visualmente verificar, os centros de consumo ficaram mais aglutinados próximos às áreas mais de maior

densidade urbana, como Vitória e Vila Velha, tornando-se mais esparsos em regiões mais distantes da capital, como Fundão e Guarapari.

As instâncias foram nomeadas de *vixnnn,* onde *nnn* representa o número de vértices (estações/consumidores), variando entre 50 e 250, em intervalos de 50 vértices, ou seja, *vix050, vix100, vix150, vix200 e vix250.*

A seguir estão relacionadas os valores utilizados para os dados de entrada requeridos pelo modelo de Baouche e colaboradores (2014). Conforme explicado na etapa 3.1, devido a eletrificação da frota brasileira de automóveis ser praticamente inexistente, foram utilizados valores típicos para alguns dados de entrada do problema, baseados em informações disponíveis nos EUA e países da Europa (EVADOPTION, 2018):

- A demanda de energia nos centros de consumo, D, associada a cada vértice foi fixada em 24 kWh, conforme explicado na etapa 3.1 (EVADOPTION, 2018).

- A energia necessária para um veículo se deslocar de um centro de consumo a uma estação de carregamento, d, foi obtida multiplicando-se os valores de distância de percurso pela energia necessária para um EV rodar a distância de um Km. Tal custo foi fixado em 0,01865 kWh/Km, conforme explicado na etapa 3.1.

- O número médio de veículos se deslocando para um centro de consumo, $n_i^{(ev)}$, foi fixado em 4, de tal forma que, para a maior das instâncias testadas, com 250 centros de consumo (vértices), se obtivesse um número total de veículos igual ao tamanho total da frota de EVs considerada neste trabalho (1000 EVs).

- A capacidade das estações de carregamento, C, foi fixada em 4.800 kWh, conforme explicado na etapa 3.1.

- O custo para se situar uma estação de carregamento no local candidato, f, foi fixado em R$ 600.000,00, conforme explicado na etapa 3.1 (AGENBROAD; HOLLAND, 2014).

Figura 23 – Código *T-SQL* para busca na *Google Maps Distance Matrix API*

```
Use GoogleMapsDB
...

-- Create table to store distance matrix from Google API
if object_id ('Distances') is null
     create table Distances (IDOrig int not null,IDDest int not null, Dist   float,
     CONSTRAINT PK_Distances PRIMARY KEY (IDOrig, IDDest))
...

-- Create OLE Automation object to send HTTP requests via SQL Server
EXEC @err1 = sp_OACreate 'MSXML2.XMLHTTP', @Obj1 OUT;
...
-- Loop through all locations determing driving distances
Declare @j int
Declare  @i int = 1;
While @i <=  (Select count(*) from PlacesRanked)
Begin
     Set @j = 1
     While @j <= (Select count(*) from PlacesRanked)
     Begin
          ...
          -- Get coordinates from Locations table
          Set @CoordOrig = (select top 1 Convert(varchar(25),Latitude) + ','+
                    Convert(varchar(25),Longitude) From PlacesRanked where ID = @i)
          Set @CoordDest = (select top 1 Convert(varchar(25),Latitude) + ','+
                    Convert(varchar(25),Longitude) From PlacesRanked where ID = @j)
          ...
          -- Request distance between @CoodOrig and @CoordDest
          SET @StatuserviceUrl =
                    'https://maps.googleapis.com/maps/api/distancematrix/xml?origins='
                    + @CoordOrig + '&destinations='                                        +
@CoordDest +'&mode=driving&language=en-EN&units=metric'
                    +'&key=AIzaSyA8csjXdiyfVewRc3TX-AExZbXc9g91PYg'

          EXEC sp_OAMethod @Obj1, 'open', NULL, 'get', @StatuserviceUrl, 'false'
          EXEC sp_OAMethod @Obj1, 'responseText', @ResponseText OUTPUT
          ...
          -- Populate distance matrix table
          SET @Distance = @ResponseText.value
                    ('(DistanceMatrixResponse/row/element/distance/value)[1]', 'float')
          INSERT INTO Distances VALUES (@i,@j,@Distance)
          ...
          Set @j = @j + 1
     End
     Set @i = @i + 1
End
...
```

Fonte: Elaborado pelo autor (2017)

- O custo do custo do quilowatt-hora multiplicado por um período de tempo T para obtenção de lucro, representado por $\propto$, foi determinado empiricamente, de tal forma que, para valores da função objetivo (3) próximos ao ótimo, o segundo termo da função objetivo ($\propto \sum_{i \in I} \sum_{j \in J} n_i^{(ev)} d_{ij}\, y_{ij}$) tivesse a mesma ordem de magnitude do primeiro termo da função ($\sum_{j \in J} f_j\ x_j$).

  Como se tratam de termos dissonantes, se $\propto$ foss muito pequeno, tornando o segundo termo pouco representativo no valor total da função objetivo, um número mínimo possível

de estações de carregamento de EVs seriam alocadas, uma vez a minimização dos custos de implantação seria o fator predominante. Por outro lado, se $\propto$ foss muito grande, a minimização da energia dispendida pelos proprietários de EVs seria preponderante, levando a alocação de um número, talvez, muito elevado de estações, e possivelmente afetando negativamente a lucratividade dos futuros proprietários de tais estações.

O valor utilizado para $\propto$ foi de 540, correspondente à um período *T* de 3 anos (1080 dias) multiplicado pelo custo aproximado do kWh no estado do Espirito Santo (R$ 0,50 / kWh) (AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA, 2017). Nas instâncias de teste, para este valor de $\propto$ (540), o custo total de instalação das estações de carregamento de EVs, correspondente ao primeiro termo da função objetivo (3), variou de 36,70% a 51,01% do valor total, dependendo da instância.

- A distância mínima entre duas estações de carregamento, foi fixada em 5 Km. É importante mencionar que a distância entre estações também foi avaliada em termos de distância de percurso entre as mesmas, obtidas através do *Google Maps Distance Matrix API*. Isto foi feito de modo a evitar que locais candidatos a estações de carregamento que estivessem fisicamente próximos um ao outro, em termos de distância em linha reta, mas separados por uma longa distância de percurso, pudessem ser considerados como intercambiáveis, em termos de atendimento a uma dada região. Por exemplo, uma estação situada em uma margem do Canal de Vitória, não deveria ser designada para atender uma região na margem oposta, a menos que houvesse uma rota relativamente curta ligando-as, como no caso de áreas próximas à 3ª ponte.

3.7.4 Formato da Base de Dados *vix*

Um programa em linguagem de programação *Transact-SQL* (T-SQL) foi desenvolvido para gerar os arquivos texto contendo os dados das instâncias *vix*. Um arquivo foi criado para cada instância, seguindo um formato semelhante ao usado para as instâncias *cpmp* da *OR-Library*, conforme descrito no item 3.4.2. A única diferença é que, as linhas que definem cada vértice foram modificadas de modo a conter, além dos dados do *cpmp* (número do vértice, coordenadas *x* e *y* e demanda), um vetor de distâncias de percurso. Cada elemento do vetor contém a distância de percurso, obtida no item 3.7.2, do vértice representado pela linha até cada um dos outros vértices. A Figura 24 ilustra as primeiras linhas do problema *vix250*.

Figura 24 – Problema *vix250*

```
1
 1 30193976.64
 250 23 2400
 1 -20.2976 -40.2957 24 0 11462 35318 7107 1868 12435 51148 12442 8604...
 2 -20.2138 -40.2667 24 12025 0 23856 17572 10050 28931 72835 32511 19070...
 3 -20.0603 -40.1895 24 35303 24310 0 42258 33867 53617 97521 57197 43756...
 4 -20.3405 -40.2855 24 5620 16138 39993 0 7598 15913 45777 15920 2135...
 5 -20.2866 -40.2974 24 2517 10011 33866 8820 0 14650 52861 14657 10318...
 6 -20.3323 -40.3772 24 12382 31653 55509 16353 15313 0 48546 1081 14311...
 7 -20.658 -40.4939 24 51086 72930 96786 45817 53064 48826 0 48811 43725...
 8 -20.3351 -40.3713 24 11314 21792 45648 15285 14245 3294 50860 0 13243...
 9 -20.3524 -40.2933 24 7360 17878 41734 2092 9339 17654 44017 17661 0...
 10 -20.3528 -40.3279 24 10631 21149 45005 5715 12609 10169 46424 8499...
 11 -20.1988 -40.2543 24 13855 2862 23026 20810 12419 32169 76073 35749...
 12 -20.3059 -40.2961 24 1080 11598 35454 5953 3398 11895 49994 11902...
 13 -20.2821 -40.2933 24 2789 9911 33229 9092 889 14922 53133 14929 10589...
 14 -20.1291 -40.3053 24 23488 14075 29171 29035 21513 40394 84298 43974...
 15 -20.2374 -40.2788 24 8828 4377 28232 14374 6852 18973 58415 18981...
```

Fonte: Elaborado pelo autor (2017)

4 RESULTADOS E DISCUSSÃO

Os resultados obtidos pela adaptação da metaheurística CRO, para os para os modelos de p-mediana e modelo FCLP Baouche, na resolução dos problemas selecionados da literatura e criados para esta dissertação, que contém distâncias reais de deslocamento, são apresentados a seguir.

Todos os algoritmos resultantes da adaptação da metaheurística CRO foram codificados na linguagem de programação C# (EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION, 2006) e executados em um PC, com sistema operacional Microsoft Windows 10, processador Intel i7 de 2,3 GHz e 16 GB de RAM.

4.1 RESULTADOS PARA A P-MEDIANA NÃO-CAPACITADA

4.1.1 Descrição dos experimentos e calibração dos parâmetros do CRO

A base de dados *pmed* da *OR-Library* (Beasley, 1990), amplamente utilizada pela comunidade científica para testes e validação de problemas de p-mediana, foi utilizada para avaliar o desempenho da implementação do CRO. A base contém 40 problemas do tipo p-mediana, com tamanhos entre 100 e 900 estações/consumidores e 5 a 100 medianas, e está descrita no item 3.4.1.

Inicialmente, cada um dos 40 problemas da *OR-Library* foi resolvido, de forma exata, utilizando o otimizador IBM CPLEX. A seguir, na mesma plataforma computacional, cada problema foi resolvido 20 vezes pelo aplicativo CRO, utilizando-se a mesma configuração para todos os problemas.

A configuração dos parâmetros operacionais do CRO utilizada na resolução dos problemas foi obtida empiricamente, na forma de tentativa e erro, como descrito a seguir:

- Inicialmente, foram atribuídos aos parâmetros operacionais do CRO os valores sugeridos por Lam e Li (2012) em seu tutorial, a saber: *PopSize = 10, KELossRate = 0,2, MoleColl = 0,2, InitialKE = 1000,* α *= 500,* β *= 10 e buffer = 0.* O número máximo de iterações foi fixado em 1.000 e o número máximo de iterações realizadas sem melhorias em 500 iterações.

- Em seguida, cada um dos parâmetros *PopSize, KELossRate, MoleColl, InitialKE,* α e β foram modificados, em busca de valores de função objetivo mais baixos. Cada parâmetro foi avaliado individualmente, fazendo-se variar o parâmetro em questão, enquanto os outros eram mantidos inalterados. Cada um dos problemas foi resolvido de 2 a 5 vezes para cada configuração diferente testada. O valor inicial do parâmetro *buffer* foi mantido em zero em todos os experimentos. A seguir estão relacionadas algumas das conclusões, obtidas empiricamente, durante o processo de calibração:

    - *PopSize*: os melhores resultados, em termos de tempos computacionais, foram obtidos com populações iniciais pequenas, entre 10 e 100 moléculas. Isto permitiu que ocorresse intensificação suficiente, ou seja, que existissem colisões unimoleculares e multimoleculares ineficazes em quantidade tal que se pudessem alcançar os mínimos locais entre as vizinhanças, sem que o número de iterações aumentasse demasiadamente.

    - *KELossRate:* O valor inicial de 0,2, sugerido por Lam e Li (2012), provou ser muito baixo para a aplicação no modelo da p-mediana não-capacitada e operadores utilizados. Nessas condições, a perda de energia cinética da molécula era muito alta, permitido que sínteses ocorressem mais rapidamente do que o desejável, sem que houvesse intensificação (busca local) suficiente das soluções armazenadas nas moléculas. Após a calibração, *KELossRate* foi fixado em *0,8* para todos os experimentos.

    - *MoleColl*: O valor 0*,2* se mostrou adequado, sendo utilizado em todos os experimentos.

    - *InitialKE*: O valor inicial de 1.000 se mostrou muito baixo, causando um elevado número de decomposições rejeitadas e sínteses prematuras. Por fim, utilizou-se um valor 100 vezes maior que o inicial ou cerca de dez vezes maior do que o valor ótimo de função objetivo mais alto dentre os 40 problemas de p-mediana avaliados.

    - α: O valor inicial de 500 se mostrou extremante elevado, praticamente impedindo que ocorressem decomposições. Dada a natureza do algoritmo *Fast Interchange*, empregado nas colisões ineficazes com a parede e multimoleculares, de obtenção da maior melhoria (*best improvement*), basta uma execução do mesmo para que se atinja o mínimo local para uma dada solução. Desta forma, chamadas subsequentes ao algoritmo para intensificação da mesma solução não iriam resultar em nenhuma melhoria. Assim, atribuiu-se o valor 1 ao parâmetro α, a fim de evitar que isto ocorresse.

- β: De forma oposta ao parâmetro α, o valor inicial atribuído ao parâmetro β se mostrou excessivamente pequeno, resultando em um número de sínteses muito baixo ou inexistente. Por fim, chegou-se a um valor de *10.000* ou 10% do valor inicial de energia cinética da molécula (*InitialKE*).

- Após a calibração dos parâmetros básicos do CRO, procedeu-se a calibração do número de iterações que, partindo de um valor inicial (1.000), foi incrementado até que a otimalidade fosse atingida ou não fosse possível obter melhorias nos resultados (*gaps* estáveis). O número de iterações sem melhorias foi mantido constante (500 iterações) em todos os experimentos.

Após a calibração, a configuração final dos parâmetros operacionais do CRO utilizada na resolução de todos os problemas foi a seguinte: *PopSize = 10, KELossRate = 0,8, MoleColl = 0,2, InitialKE = 100.000,* α *= 1,* β *= 10.000 e buffer = 0.* Como critério de parada, limitou-se o número máximo de iterações em 5.000 e o número máximo de iterações realizadas sem melhorias em 500 iterações.

4.1.2 Resultados

Conforme mostra a Tabela 4, o CRO atingiu o valor ótimo da função objetivo em 35 dos 40 problemas da biblioteca *OR-Library*, sendo que o melhor *gap* foi de apenas 0,25% na resolução do problema *pmed36*. Os demais *gaps* ficaram abaixo de 0,06%.

Neste trabalho, *gap* é definido como a diferença percentual entre o resultado obtido pelo método heurístico, exato ou híbrido e o seu respectivo valor ótimo (ou melhor valor conhecido), na resolução de um problema. Adicionalmente, o melhor *gap* é definido como a diferença percentual entre o menor valor da função objetivo obtido pelo método heurístico, exato ou híbrido e o seu respectivo valor ótimo (ou melhor valor conhecido).

Uma vez que a mesma plataforma computacional foi utilizada para resolver os problemas, de forma exata, pelo CPLEX e pelo aplicativo que implementa o CRO, uma comparação precisa de desempenho (tempos de execução) entre os dois métodos pôde ser efetuada. Na comparação com outras metaheurísticas, foi efetuada somente uma avaliação de *gaps*, uma vez que hardware, linguagens de programação e sistemas operacionais nos quais os diferentes algoritmos foram implementados, diferem significativamente.

Na comparação com os resultados obtidos na resolução pelo otimizador CPLEX, para os problemas menores (*pmed01* a *pmed20*), o CRO foi cerca de duas vezes mais rápido, com uma média de 12,14s contra 24,63s do CPLEX, conforme mostra a Tabela 4. Para os problemas de maior dimensão (*pmed21* a *pmed40*), a média dos tempos de execução do CRO foi de 124,1s, contra 8157,68s do CPLEX. A média geral de tempos de execução do CRO foi de 68,14s, contra 4091,16s do CPLEX, portanto, cerca de 60 vezes mais rápida que as obtidas pelo otimizador da IBM. Entretanto, em 13 dos problemas avaliados o CPLEX apresentou tempos de execução menores, especialmente naqueles em que o tempo total para obtenção dos valores ótimos não excedeu a 3 minutos e a relação entre o número de medianas, *p,* e o número de vértices era baixo, entre 3 e 5, na maioria dos casos. Contudo, a maior diferença não excedeu a 101s, ficando, na média, em 21,19s.

Os resultados obtidos pelo CRO também foram comparados com aqueles obtidos por Daskin e Maass (2015) na resolução dos mesmos 40 problemas da *OR-Library*, utilizando Relaxação Lagrangiana (RL). Embora os resultados da resolução por esta técnica tenham permitido atingir valores ótimos em 39 dos 40 problemas, no problema *pmed40* o melhor *gap* foi de 19,5%, versus 0,04% do CRO. Para a resolução do problema *pmed36* o tempo de resolução foi de aproximadamente 768s, contra 43s requeridos pelo CRO. O tempo médio de execução do CRO para todos os problemas foi de 68,14s, ligeiramente mais lento do que os 44,87s obtidos pela relaxação Lagrangiana. No entanto, o CRO provou ser mais rápido em 18 dos 40 problemas. Deve-se notar que esta comparação de desempenho é apenas aproximada, uma vez que as plataformas computacionais são semelhantes, mas não idênticas.

Os resultados do CRO também foram comparados com aqueles obtidos na resolução de 22 problemas *pmed* da biblioteca *OR-Library*, por Hansen e Mladenović (1997), utilizando a metaheurística *Variable Neighborhood Search* (VNS). Nos problemas 15,25,29 e 30, os melhores *gaps* obtidos pelo VNS foram ligeiramente menores que os do CRO, tendo o maior deles ocorrido no problema *pmed30*: 0,15% do VNS versus 0,25% do CRO. Para o problema *pmed40*, o *gap* foi exatamente o mesmo: 0,04%. O CRO foi ligeiramente melhor nos problemas 14 e 19. Ambos CRO e VNS atingiram valores idênticos em 16 dos 22 problemas.

Por fim, os resultados foram comparados com os obtidos por Alp, Erkut e Drezner (2003), que propuseram uma versão eficiente de Algoritmo Genético (GA) para a resolução do problema da p-mediana não-capacitada. Os autores tiveram sucesso em atingir o valor ótimo da função objetivo em somente 28 dos 40 problemas *pmed* da *OR-Library*. Em todos os problemas, a

metaheurística GA apresentou melhores *gaps* maiores que os obtidos pelo CRO. A média dos melhores *gaps* do CRO foi de 0,0108% contra 0,0363% do GA, portanto, 3,36 vezes maior que a obtida pelo GA. Em seu trabalho, Alp, Erkut e Drezner (2003) também forneceram o desvio médio para todos os problemas resolvidos por algoritmo GA. Sendo assim, foi possível comparar os resultados com aqueles obtidos pelo CRO. A somatória dos desvios médios do GA foi de 4,40% contra 2,61% do CRO.

Os resultados obtidos pelo otimizador CPLEX, bem como aqueles obtidos na resolução por CRO, RL, VNS e GA estão sumarizados na Tabela 4. Para todas as técnicas são apresentados o melhor *gap*, exceto para o CPLEX, já que os valores da função objetivo são sempre os ótimos. Para o CRO e GA estão listados também o desvio médio para os diferentes problemas *pmed* da *OR-Library*.

Tabela 4 – Comparação do CRO com outras heurísticas para as instâncias *pmed*

	OR-Library			*CPLEX*	*CRO*			*RL*		*VNS*	*GA*	
Pmed	***p***	***Tam***	***Valor ótimo (Z)***	***Tempo Exec. (s)***	***Melhor gap (%)***	***Desvio Med. (%)***	***Tempo Med. Exec (s)***	***Melhor gap (%)***	***Tempo Med. Exec (s)***	***Melhor gap (%)***	***Melhor gap (%)***	***Desvio Médio (%)***
1	5	100	5819	0,39	0,00	0,00	0,40	0,00	2,94	-	0,00	0,14
2	10	100	4093	0,75	0,00	0,03	0,53	0,00	8,92	0,00	0,00	0,20
3	10	100	4250	0,50	0,00	0,00	0,48	0,00	7,70	-	0,00	0,33
4	20	100	3034	0,33	0,00	0,03	0,77	0,00	3,06	0,00	0,00	0,29
5	33	100	1355	0,31	0,00	0,00	0,91	0,00	3,03	-	0,00	0,00
6	5	200	7824	6,33	0,00	0,00	1,20	0,00	15,09	-	0,00	0,03
7	10	200	5631	3,31	0,00	0,01	1,91	0,00	3,08	0,00	0,00	0,21
8	20	200	4445	2,61	0,00	0,00	3,37	0,00	3,09	0,00	0,00	0,00
9	40	200	2734	2,56	0,00	0,07	5,68	0,00	14,73	0,00	0,00	0,30
10	67	200	1255	2,36	0,00	0,03	6,99	0,00	5,31	0,00	0,08	0,29
11	5	300	7696	28,53	0,00	0,00	3,54	0,00	4,81	-	0,00	0,16
12	10	300	6634	21,97	0,00	0,00	4,41	0,00	17,30	-	0,00	0,03
13	30	300	4374	15,45	0,00	0,01	11,54	0,00	4,80	-	0,00	0,00
14	60	300	2968	18,86	0,00	0,06	21,60	0,00	8,70	0,03	0,00	0,01
15	100	300	1729	17,02	0,06	0,26	24,62	0,00	7,94	0,00	0,23	0,39
16	5	400	8162	133,77	0,00	0,00	6,34	0,00	24,55	-	0,00	0,03
17	10	400	6999	81,94	0,00	0,05	8,68	0,00	27,89	-	0,00	0,01
18	40	400	4809	57,42	0,00	0,04	28,92	0,00	6,55	0,00	0,00	0,00
19	80	400	2845	51,03	0,00	0,13	49,67	0,00	9,30	0,04	0,04	0,12
20	133	400	1789	47,19	0,00	0,11	61,31	0,00	24,50	0,00	0,17	0,22
21	5	500	9138	109,27	0,00	0,00	9,33	0,00	3,70	-	0,00	0,09
22	10	500	8579	3023,91	0,00	0,20	16,34	0,00	55,86	0,00	0,00	0,00
23	50	500	4619	105,33	0,00	0,02	73,30	0,00	8,64	-	0,00	0,05
24	100	500	2961	106,20	0,00	0,09	103,72	0,00	41,42	0,00	0,03	0,10
25	167	500	1828	106,41	0,05	0,39	123,18	0,00	72,44	0,00	0,22	0,27
26	5	600	9917	9243,08	0,00	0,00	14,58	0,00	22,25	-	0,00	0,01
27	10	600	8307	310,69	0,00	0,00	25,13	0,00	12,53	-	0,00	0,04
28	60	600	4498	181,13	0,00	0,05	137,54	0,00	12,30	0,00	0,02	0,02
29	120	600	3033	164,30	0,03	0,09	215,48	0,00	18,81	0,00	0,07	0,13
30	200	600	1989	162,77	0,25	0,60	235,73	0,00	57,55	0,15	0,40	0,48
31	5	700	10086	639,58	0,00	0,00	19,93	0,00	29,00	-	0,00	0,03
32	10	700	9297	15721,33	0,00	0,00	32,93	0,00	15,41	0,00	0,00	0,02
33	70	700	4700	264,66	0,00	0,08	194,94	0,00	19,88	0,00	0,00	0,04
34	140	700	3013	239,06	0,00	0,14	339,65	0,00	33,02	0,00	0,07	0,09
35	5	800	10400	31208,52	0,00	0,00	26,51	0,00	47,64	-	0,00	0,03
36	10	800	9934	35782,86	0,00	0,00	43,22	0,00	767,16	-	0,00	0,00
37	80	800	5057	352,31	0,00	0,04	310,32	0,00	97,06	0,00	0,02	0,05
38	5	900	11060	61317,58	0,00	0,00	35,98	0,00	107,78	-	0,00	0,09
39	10	900	9423	3595,95	0,00	0,00	50,21	0,00	136,27	-	0,00	0,00
40	90	900	5128	518,69	0,04	0,09	474,66	19,50	32,89	0,04	0,10	0,10

Fonte: Elaborado pelo autor (2017)

4.2 RESULTADOS PARA A P-MEDIANA CAPACITADA

4.2.1 Descrição dos experimentos

Para a avaliação da adaptação da metaheurística CRO para a p-mediana capacitada (CPMP), um número de experimentos computacionais foi realizado utilizando bases de dados de referência obtidas da literatura. Foram utilizados três conjuntos de instâncias diferentes, com número de vértices variando de 100 a 724 e número de medianas variando de 5 a 300.

O primeiro conjunto de instâncias foi proposto por Osman e Christofides (1994) e tem sido amplamente utilizada em comparações de desempenho de soluções para o CPMP. O primeiro grupo de 10 instâncias deste conjunto de dados, denominado *cpmp01* a *cpmp10*, possui 50 vértices e 5 medianas, enquanto o último grupo, também com 10 instâncias, denominado *cpmp11* a *cpmp20*, possui 100 vértices e 10 medianas. Os resultados da adaptação do CRO para o CPMP foram comparados termos de *gaps* com os obtidos por Osman e Christofides (1994), utilizando uma metaheurística híbrida (SATS) contendo elementos de *Simulated Annealing* e Busca Tabu. Além disso, compararam-se os *gaps* e tempos computacionais do CRO com os resultados obtidos na solução das mesmas instância usando o IBM CPLEX vs. 12.6 no mesmo hardware do CRO e, por último, com o método mateurístico IRMA, proposto por Stefanello e colaboradores (2015), rodando em hardware e otimizador MIP semelhantes (Intel i5 e CPLEX 12.3).

O segundo conjunto, proposto por Lorena & Senne (2004), é composto por seis instâncias, denominadas de *sjc1* a *sjc4b*, construídas a partir de dados coletados de um banco de dados geográfico da cidade de São José dos Campos, Brasil. O número de vértices varia de 100 a 400 e o número de medianas de 10 a 40. Compararam-se *gaps* e tempos computacionais da adaptação do CRO para o CPMP com os seguintes trabalhos recentes da literatura:

- Heurística *Scatter Search* (SS) desenvolvida por Scheuerer e Wendolsky (2006).
- Heurística *Variable Neighborhood Search* (VNS) combinada com CPLEX, por Fleszae e Hindi (2008)
- Heurística *Clustering Search* (CS) apresentada por Chaves e colaboradores (2007)
- Método híbrido (*Fen-CPLEX*) que alia técnica *Fenchel cutting planes* com CPLEX, por Boccia e colaboradores (2008).

- Mateurística IRMA, proposta por Stefanello e colaboradores (2015).

Além disso, *gaps* e tempos computacionais do CRO para CPMP foram comparados com os resultados obtidos por Stefanello e colaboradores (2015) na solução das mesmas instâncias usando IBM CPLEX vs. 12.3, em hardware similar.

Os últimos dos três conjuntos que foram utilizados para comparações de *gaps* e tempo computacional foram propostos por Stefanello e colaboradores (2015) e consistem numa adaptação de problemas da TSP-LIB, com número de vértices variando de 318 a 724 e número de medianas entre 5 e 200. Tais instâncias foram selecionadas, não somente por serem recentes, mas por que a maioria delas não pode ser resolvida em tempo hábil por otimizadores MIP, como o CPLEX, o que justifica o uso de métodos heurísticos.

Após efetuar a calibração adequada dos parâmetros do CRO, todas as instâncias foram resolvidas 20 vezes, registrando-se tempos de execução e valores da função objetivo. O melhor valor da função objetivo obtido e o desvio padrão também foram registrados. Além de resolver as instâncias usando o CRO para o CPMP, o primeiro conjunto de instâncias também foi resolvido pelo CPLEX vs. 12.6, sendo executado no mesmo hardware.

4.2.2 Calibração dos Parâmetros do CRO

Os procedimentos de calibração dos parâmetros do CRO foram desenvolvidos com base na premissa de que uma boa heurística deve fornecer resultados aceitáveis, isto é, valores ótimos ou próximos deste, em tempos computacionais baixos. Desta forma, durante os testes, foram privilegiadas as configurações que permitissem atingir tempos de execução mais baixos em detrimento das que visassem atingir a otimalidade.

Adicionalmente, sempre que possível, tentou-se atingir um *gap* médio de menos de 1% nas instâncias testadas. No presente trabalho, *gap* é definido como como a diferença percentual entre os valores da função objetivo obtidos pelo CRO, ou qualquer outra metaheurística ou método exato, e o melhor valor de função objetivo conhecido para uma determinada instância.

O CRO é uma metaheurística altamente parametrizada. Existem sete parâmetros especificados em Lam e Li (2012) que podem afetar *gaps*, tempos de execução ou ambos. Esses parâmetros são *PopSize, KELossRate, MoleColl, buffer, InitialKE,* α e β.

Na presente implementação foram introduzidos alguns outros parâmetros que podem afetar o desempenho. Os mesmos têm como objetivo fornecer critérios de parada (*MaxIterations* e *MaxIterationsWithoutImprovement*) e controlar o grau de intensificação de soluções durante a execução do mecanismo *λ-Interchange* (*λ* e *Minλ-InterchangeIterations*). Os tamanhos mínimo e máximo da população de moléculas são controlados pelos parâmetros *MinMol* e *MaxMol*, respectivamente. Finalmente, *κ0* e *Δκ* controlam o tamanho das listas do Conjunto de Listas de Proximidade (CLP).

Existem, portanto, um total de 15 parâmetros que podem afetar o desempenho do CRO. Uma vez que a combinação de todos estes parâmetros existe em um espaço de quinze dimensões, é impraticável testar todas as combinações possíveis. Desta forma, os parâmetros foram calibrados, utilizando uma metodologia de tentativa e erro.

Segue-se uma breve discussão sobre a metodologia que foi utilizada na calibração, além de algumas descobertas empíricas que podem ajudar em futuras implementações similares do CRO.

Foi determinando, de forma empírica, que os melhores resultados, em termos de *gaps*, são obtidos quando o número de sínteses e decomposições é inferior a 10% do número total de colisões. Isso está de acordo com Lam e Li (2012), que afirmam que "a decomposição e a síntese trazem diversificação ao algoritmo. A diversificação não pode ocorrer com muita frequência, ou o CRO se tornará um algoritmo completamente aleatório".

Também foi observado que, para um determinado valor de *KELossRate, MoleColl e InitialKE*, o número de decomposições é altamente dependente do parâmetro α. Se α é muito baixo (por exemplo, menos do que dois), moléculas carregando soluções, podem não ter tempo suficiente (expresso em número de colisões) para atingir os mínimos locais de suas soluções. Por outro lado, se α é muito alto, as decomposições podem nunca ocorrer.

Da mesma forma, o número de sínteses é altamente dependente do parâmetro *β*. Se estiver muito próximo de *InitialKE*, as sínteses ocorrerão com mais frequência do que o desejado, prejudicando assim a intensificação das soluções. Da mesma forma, se *β* é muito baixo, o número de sínteses pode não ser suficiente para prover a diversificação necessária para se escapar dos mínimos locais.

Outra consideração importante ao escolher valores apropriados para os vários parâmetros do CRO é o controle populacional. Se o número de sínteses e decomposições for muito desbalanceado, a população atingirá rapidamente *MinMol* ou *MaxMol*.

O valor atribuído ao parâmetro *InitialKE*, que determina a quantidade de energia cinética inicial de uma molécula, foi de cerca de 10 a 20 vezes maior do que o melhor valor conhecido da função objetivo, para os problemas testados. Quando *InitialKE* é muito baixo, a maioria das decomposições pode acabar sendo rejeitada, devido a uma menor tolerância em aceitar soluções de saída piores que as de entrada. Da mesma forma, as sínteses podem começar a ocorrer cedo demais, prejudicando assim a intensificação das soluções.

Encontrar um valor adequado para *MoleColl* também provou ser muito importante. O valor 0,1 foi escolhido para todos os experimentos, o que significa que a probabilidade de colisões intermoleculares acontecer é de 10%. Como o número de colisões com a parede é muito maior do que as intermoleculares, uma otimização *1-opt* ($\lambda = 1$ ou *1-interchange*) foi sempre executada para as mesmas. Isso provou ser adequado para atingir a otimalidade em várias instâncias. Para colisões intermoleculares, otimizações *1-opt* foram executadas, preferencialmente. Caso não fosse possível alcançar resultados satisfatórios, então otimizações *2-opt* foram testadas, numa tentativa de diminuir *gaps* ou desvios padrão. Executar uma lógica *2-interchange* requer consideravelmente mais tempo, pois o número de comparações aumenta substancialmente. Além disso, a otimização *2-opt* deve ser executada para cada uma das duas moléculas envolvidas em uma colisão intermolecular. Utilizar valores maiores do que 2 para otimizações λ*-opt* mostrou-se impraticável devido ao aumento dramático nos tempos de execução.

Começando com valores típicos sugeridos em Lam e Li (2012), em seu tutorial do CRO, e utilizando uma metodologia de tentativa e erro, determinou-se que os valores iniciais para os seguintes parâmetros padrão do CRO funcionaram bem com a maioria das instâncias testadas: *PopSize = 10, KELossRate = 0,8, MoleColl = 0,1, InitialKE = 1.000.000,* $\alpha = 10$, $\beta = 100.000$ *e buffer = 0.*

Utilizando-se o mesmo processo de calibração, definiram-se os valores típicos para os parâmetros adicionais da presente implementação do CRO para o CPMP, como: *MinMol = 2, MaxMol = 100, Min*λ*-InterchangeIterations = 1,* $\kappa 0 = 1$ *e* $\Delta\kappa = 1$.

Para ajustar os parâmetros que fornecem critérios de parada (*MaxIterations* e *MaxIterationsWithoutImprovement*) resolveu-se cada uma das instâncias 20 vezes usando o CRO, partindo de um valor pequeno para *MaxIterations* e *MaxIterationsWithoutImprovement*, como, por exemplo, 50 iterações. Se um *gap* médio de menos de 1% fosse obtido e pelo menos uma das execuções atingisse o melhor valor conhecido para a função objetivo, tal número de iterações seria utilizado para *MaxIterationsWithoutImprovement*, sendo *MaxIterations* configurado como o dobro do mesmo. Caso contrário, continuou-se aumentando *MaxIterations* e *MaxIterationsWithoutImprovement* em pequenos incrementos, como 50 ou 100 iterações, até que um *gap* de menos de 1% com uma execução atingindo o melhor valor conhecido fosse obtido ou não houvesse melhora significativa na função objetivo. Se esses critérios não puderem ser atendidos, a execução com a menor *gap* médio seria a escolhida.

Além disso, primeiro tentou-se resolver uma dada instância executando-se uma lógica *1-interchange* para todas as colisões ineficazes com a parede e intermoleculares. Se o *gap* desejado não pusesse ser obtido, então uma lógica *2-interchange* seria utilizada, mas somente nas colisões intermoleculares. Utilizou-se primeiramente a lógica *1-interchange*, a fim de evitar a penalidade de desempenho mencionada anteriormente.

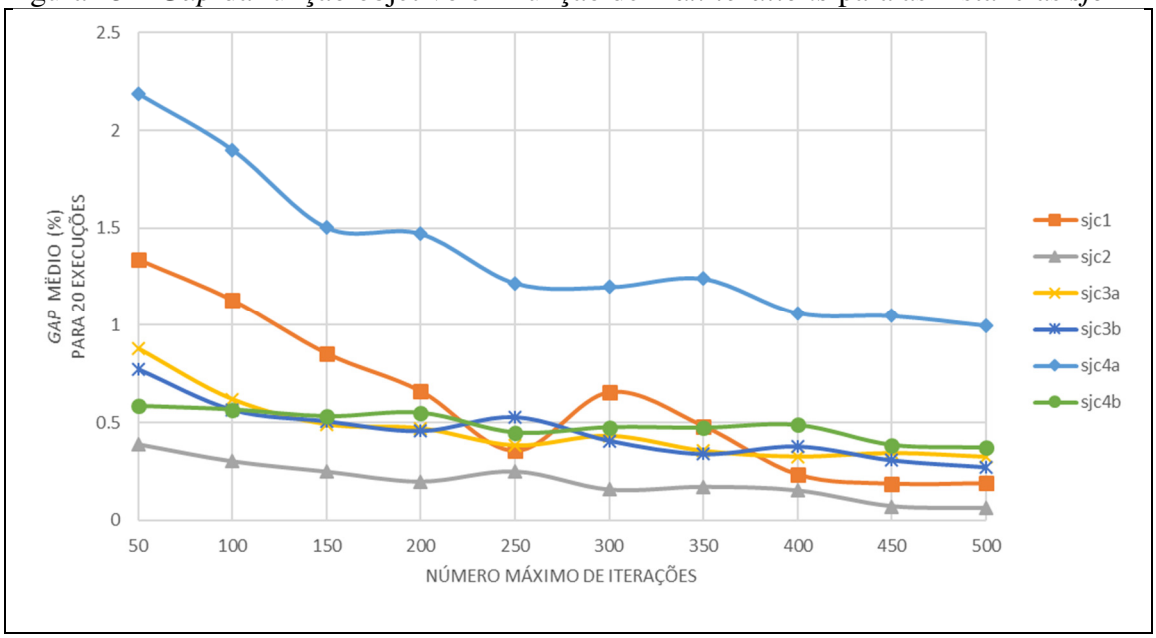
Uma vez que valores adequados para *MaxIterations* e *MaxIterationsWithoutImprovement* fossem encontrados, cada instância foi resolvida de 10 a 30 vezes, dependendo da base de dados, e os resultados utilizados na comparação com outras metaheurísticas, métodos híbridos e exatos.

É importante mencionar que muitas outras metodologias de calibração podem ser usadas para ajustar os parâmetros do CRO e que a metodologia descrita é apenas uma que funcionou adequadamente para as instâncias testadas, gerando de boas a ótimas soluções para a maioria. Contudo, é possível que existam outras metodologias que permitam gerar soluções melhores ou mais rapidamente. Portanto, o autor não afirma que a metodologia adotada seja "ótima", ou melhor do que qualquer outra, em qualquer aspecto.

Como exemplo da metodologia de calibração adotada, a Figura 25 mostra o *gap* da função objetivo em função de *MaxIterations*, para a base de dados *sjc*, proposta por Lorena e Senne (2004), composta de seis instâncias denominadas *sjc1, sjc2, sjc3a , sjc3b, sjc4a e sjc4b*. O mesmo valor foi utilizado para *MaxIterations* e *MaxIterationsWithoutImprovement* em todas as execuções. *MaxIterations* varia de 50 a 500 iterações. Os seguintes parâmetros do CRO são

fixos: *PopSize = 10, KeLossRate = 0,8, MoleColl = 1, InitialKe = 1.000.000, α = 10, β = 100.000, MinMol = 2, MaxMol = 20 e Minλ-InterchangeIterations = 1.* Uma otimização *1-opt* (*λ-interchange* = 1), foi executada para todas as instâncias, exceto na instância *sjc4a*, em que uma otimização *2-opt* foi executada para todas as colisões intermoleculares. Para a mesma base de dados, a Tabela 5 mostra o nome da instância (Inst.), número de vértices (*n*), número de medianas (*p*), *MaxIterations*, *MaxIterationsWithoutImprovement*, número de execuções (*exec.*), valor ótimo para a função objetivo (*Z*), além do melhor objetivo alcançado pelo CRO, o *gap* percentual médio (*gap*) e o desvio padrão para 20 execuções de cada combinação de instância e *MaxIterations*. As linhas que atendem aos critérios de calibração descritos neste item estão marcadas em negrito.

Figura 25 – *Gap* da função objetivo em função de *MaxIterations* para as instâncias *sjc*



Fonte: Elaborado pelo autor (2017)

Tabela 5 – Calibração do número de iterações para as instâncias *sjc*

Inst.	*n*	*p*	*Max Iterations*	*Max Iterations without Impr.*	Num. Exec.	Z	CRO		
							Objetivo Mínimo	*Gap* médio (%)	Desvio Padrão
sjc1	100	10	50	50	20	17288,99	17393,94	1,34	147,86
sjc1	100	10	100	100	20	17288,99	17288,99	1,13	137,96
sjc1	**100**	**10**	**150**	**150**	**20**	**17288,99**	**17288,99**	**0,85**	**157,41**
sjc2	**200**	**15**	**50**	**50**	**20**	**33270,94**	**33270,94**	**0,39**	**87,51**
sjc3a	300	25	50	50	20	45335,16	45480,07	0,88	151,91
sjc3a	300	25	100	100	20	45335,16	45423,66	0,62	117,61
sjc3a	**300**	**25**	**150**	**150**	**20**	**45335,16**	**45335,16**	**0,49**	**107,97**
sjc3b	300	30	50	50	20	40635,90	40669,59	0,77	148,01
sjc3b	300	30	100	100	20	40635,90	40715,97	0,57	81,50
sjc3b	300	30	150	150	20	40635,90	40689,03	0,51	103,82
sjc3b	300	30	200	200	20	40635,90	40714,96	0,46	86,73
sjc3b	**300**	**30**	**250**	**250**	**20**	**40635,90**	**40635,90**	**0,53**	**100,04**
sjc4a	402	30	50	50	20	61925,51	62626,99	2,19	338,45
sjc4a	402	30	100	100	20	61925,51	62527,66	1,90	301,87
sjc4a	402	30	150	150	20	61925,51	62460,81	1,50	213,19
sjc4a	402	30	200	200	20	61925,51	62414,62	1,47	260,53
sjc4a	402	30	250	250	20	61925,51	62351,35	1,22	130,20
sjc4a	402	30	300	300	20	61925,51	62330,35	1,20	185,05
sjc4a	402	30	350	350	20	61925,51	62403,76	1,24	215,55
sjc4a	402	30	400	400	20	61925,51	62240,75	1,06	199,39
sjc4a	402	30	450	450	20	61925,51	62311,56	1,05	162,92
sjc4a	**402**	**30**	**500**	**500**	**20**	**61925,51**	**62330,35**	**1,00**	**126,85**
sjc4b	**402**	**40**	**50**	**50**	**20**	**52458,00**	**52495,56**	**0,58**	**110,78**

Fonte: Elaborado pelo autor (2017)

4.2.3 Resultados

4.2.3.1 Instâncias cpmp da OR-Library

A primeira base de dados, composta por 20 instâncias propostas por Osman e Christofides (1994), foi resolvida utilizando-se o otimizador CPLEX vs. 12.6 e o CRO para CPMP. Inicialmente, cada instância foi resolvida à otimalidade pelo CPLEX. A Tabela 6 mostra, para cada instância, o número de vértices (n), o número de medianas (p), o valor ótimo da função objetivo (Z) , bem como o respectivo tempo de execução para resolvê-lo usando o CPLEX.

Em seguida, o CRO foi utilizado para resolver cada instância 30 vezes. O algoritmo principal foi configurado para terminar quando *MaxIterations* ou *MaxIterationsWithoutImprovement* fosse atingido. Adicionalmente, também foi configurado para terminar se o valor ótimo para a função objetivo fosse atingido em qualquer iteração.

Para cada instância de teste, foram calculadas a percentagem de vezes que a otimalidade foi alcançada, o *gap* médio, o desvio padrão, o tempo médio de execução, o número médio de iterações, decomposições, colisões com a parede, sínteses e colisões intermoleculares. Estes resultados também são apresentados na Tabela 6, bem como os valores definidos para *MaxIterations*, *MaxIterationsWithoutImprovement* e *PopSize*. Todos os outros parâmetros do CRO permaneceram constantes para todas as instâncias: *KELossRate = 0,8, MoleColl = 0,1, InitialKE = 1.000.000,* $\alpha = 10$, $\beta = 50.000$, *buffer = 0, MinMol = 2, MaxMol = 100*, *Min*λ*-InterchangeIterations = 1,* $\kappa 0 = 1$ *e* $\Delta\kappa = 1$. Uma otimização *1-opt* foi usada em todas as colisões ineficazes.

Conforme mostra a Tabela 6, o CRO atingiu, pelo menos duas vezes, o valor ótimo em todas as instâncias de teste; sendo o pior *gap* de 0,62% para a instância *cpmp11*. A média de todos os *gaps* foi de 0,089%. O algoritmo construtivo foi capaz de gerar uma solução ótima para as instâncias *cpmp02* e *cpmp04*. Portanto, o número médio de iterações para as duas instâncias é zero.

Em média, o CRO foi 18,24s mais rápido que o CPLEX, com um tempo médio de execução de 2,62s versus 20,86s do CPLEX. O CRO foi um pouco mais lento ao resolver as instâncias 11, 14 e 17. No entanto, a diferenças entre tempos de execução não excederam a 1,7s, conforme mostrado na Tabela 6.

Os resultados do CRO para o CPMP também foram comparados com aqueles reportados, para o mesmo conjunto de instâncias, por Osman e Christofides (1994), que propuseram uma metaheurística para o CPMP que envolve Simulated Annealing e Tabu Search (SATS), conforme mostrado na Tabela 7. Os mesmos autores também implementaram a metaheurística Simulated Annealing (SA), proposta por Connoly (1992), e uma metaheurística simples de Busca Tabu, proposta por Glover (1986), todas para o CPMP. Conforme mostrado na Tabela 7, o SATS atingiu valores ótimos em 17 de 20 instâncias; com o pior *gap* de 0,049%, para instância *cpmp17*. O SA conseguiu resultados ótimos em apenas 14 instâncias com o pior intervalo de 2,94% obtido na primeira instância (*cpmp01*). TS foi o pior deste grupo, com apenas 10 instâncias resolvidas à otimalidade e *gaps* de até 2,94% (*cpmp01*). Uma vez que o CRO para o CPMP alcançou valores ótimos em todas as instâncias de teste, pode-se concluir que o mesmo superou as outras metaheurísticas deste grupo. Não foram comparados tempos de execução, pois as plataformas de hardware do CRO e as usadas na implementação das metaheurísticas SATS, SA e TS diferem significativamente.

Finalmente, os resultados obtidos pelo CRO foram comparados com o método mateurístico IRMA, proposto por Stefanello e colaboradores (2015), que combina metaheurísticas com programação matemática para resolver o problema da p-mediana capacitada, conforme mostra a Tabela 7. A fim de efetuar uma comparação justa entre CRO e IRMA, dos vários resultados do IRMA disponíveis para as instâncias *cpmp*, foram escolhidos aqueles que atingiram valores ótimos em todas as instâncias de teste (identificados por IRMA $\alpha = 2.4$), de modo que os resultados fossem equivalentes à presente implementação do CRO, em termos de *gaps*.

Como o IRMA foi testado em hardware moderno (Intel i5-2300 CPU de 2.8 GHz com 4 GB de RAM), uma comparação mais justa de tempos de execução pôde ser feita. Assim, o CRO foi mais rápido que IRMA em metade das instâncias testadas, como mostrado na Tabela 7. A média dos tempos de execução do IRMA foi de 4,71s contra 2,62s do CRO, tornando-o ligeiramente mais rápido.

4.2.3.2 Instâncias sjc

O segundo conjunto de instâncias usado para avaliar a adaptação do CRO para o CPMP é composto de 6 instâncias propostas por Lorena e Senne (2004). Todos as instâncias foram resolvidas 30 vezes pelo CRO e, então, os resultados foram comparados, em termos de *gaps* e tempos de execução, com os métodos heurísticos e exatos descritos no item 4.2.1.

Incialmente, os resultados do CRO foram comparamos em termos de *gaps* e tempos de execução com os obtidos por Stefanello e colaboradores (2015), utilizando a mateurística IRMA, que combina uma heurística de redução de modelos e um otimizador MIP (CPLEX 12.3), conforme mostra a Tabela 9 . Em seu trabalho, Stefanello também resolveu todas as instâncias *sjc* usando o CPLEX, sem auxílio do IRMA. Tais resultados também foram utilizados em comparações com o CRO, já que o hardware e versão do CPLEX são semelhantes.

Adicionalmente, os resultados do CRO foram comparados com outros trabalhos recentes disponíveis na literatura, conforme descrito no item 4.2.1. São eles: *Scatter Search* (SCHEUERER; WENDOLSKY, 2006), *Variable Neighborhood Search* (FLESZAR; HINDI, 2008), *Clustering Search* (CHAVES et al., 2007) e Fen-CPLEX (BOCCIA et al., 2008), conforme mostra a Tabela 9. É importante notar que, apesar de serem trabalhos relativamente recentes, as plataformas de hardware utilizadas pelos autores podem ser consideradas obsoletas, consistindo de um processador Intel Celeron de 2.2 GHz de baixo desempenho, um processador Intel Pentium IV 3.2 GHz, um processador Intel Pentium IV 3.02. GHz e um processador de l.6 GHz para laptop, respectivamente. Portanto, uma comparação de tempos de execução é fornecida apenas para fins de referência, pois não seria justo comparar os resultados do CRO com os publicados pelos autores supracitados, considerando que estes processadores foram lançados há mais de uma década. Uma outra razão é que os tempos de execução reportados pelos autores são, em média, de 28 a 197 vezes mais lentos que os tempos reportados por Stefanello e colaboradores (2015), tornando uma comparação de tempos de execução menos relevante.

Tabela 6 – Resultados computacionais do CRO para as instâncias *cpmp*

INSTÂNCIA			**CPLEX 12.6**		**CRO**											
ID	***n***	***p***	***Z***	**Tempo Exec. (s)**	**Otima-lidade Atingida (%)**	***Gap Médio* (%)**	**Desvio Padrão**	**Tempo Médio Exec. (s)**	**Média Iter.**	**Média Decomp.**	**Média Col. Parede**	**Média Sint.**	**Média Col. Inter.**	**Max Iter.**	***Max. Iter. w/o Impr.***	***Pop Size***
cpmp01	50	5	713	0,53	100,00	0,00	0,00	0,12	1	0	0	0	0	1000	500	10
cpmp02	50	5	740	0,13	100,00	0,00	0,00	0,12	0	0	0	0	0	1000	500	10
cpmp03	50	5	751	0,33	100,00	0,00	0,00	0,14	18	0	16	0	2	1000	500	10
cpmp04	50	5	651	0,25	100,00	0,00	0,00	0,12	0	0	0	0	0	1000	500	10
cpmp05	50	5	664	0,35	100,00	0,00	0,00	0,13	5	0	5	0	0	1000	500	10
cpmp06	50	5	778	0,25	100,00	0,00	0,00	0,15	10	0	9	0	0	1000	500	10
cpmp07	50	5	787	0,80	100,00	0,00	0,00	0,15	1	0	1	0	0	1000	500	10
cpmp08	50	5	820	3,02	16,67	0,21	0,87	1,86	595	17	514	18	43	1000	500	10
cpmp09	50	5	715	0,39	100,00	0,00	0,00	0,16	11	0	9	0	1	1000	500	10
cpmp10	50	5	829	2,70	76,67	0,13	2,55	0,99	319	10	275	7	26	1000	500	10
cpmp11	100	10	1006	3,77	20,00	0,62	3,63	4,99	500	12	433	13	40	1000	500	10
cpmp12	100	10	966	3,96	100,00	0,00	0,00	1,09	113	2	99	0	11	1000	500	10
cpmp13	100	10	1026	1,20	100,00	0,00	0,00	0,96	89	0	79	0	8	1000	500	10
cpmp14	100	10	982	5,47	6,67	0,13	1,14	7,15	712	18	618	23	46	1000	500	10
cpmp15	100	10	1091	5,52	10,00	0,08	0,31	5,10	533	15	460	15	41	1000	500	10
cpmp16	100	10	954	3,41	100,00	0,00	0,00	1,01	100	1	87	0	10	1000	500	10
cpmp17	100	10	1034	5,27	26,67	0,25	4,00	5,86	562	13	486	17	41	1000	500	10
cpmp18	100	10	1043	5,50	100,00	0,00	0,00	0,65	60	0	53	0	6	1000	500	10
cpmp19	100	10	1031	4,80	86,67	0,03	0,74	3,14	297	6	258	6	24	1000	500	10
cpmp20	100	10	1005	369,61	13,33	0,33	2,91	18,43	1831	39	1593	53	123	2000	2000	20
Médias				**20,86**	**72,83**	**0,09**		**2,62**								

Fonte: Elaborado pelo autor (2017)

Tabela 7 – Comparação do CRO com outras heurísticas para as instâncias *cpmp*

Instância		**CRO**		**IRMA (α = 2.4)**		**SATS**	**SA**	**TS**
ID	**Z**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Melhor gap (%)**	**Melhor *gap* (%)**
cpmp01	713	0,00	0,11	0,00	0,20	0,00	2,94	2,94
cpmp02	740	0,00	0,11	0,00	0,05	0,00	0,00	0,00
cpmp03	751	0,00	0,12	0,00	0,16	0,00	0,00	0,00
cpmp04	651	0,00	0,11	0,00	0,08	0,00	0,00	0,00
cpmp05	664	0,00	0,12	0,00	0,15	0,00	0,00	0,00
cpmp06	778	0,00	0,12	0,00	0,08	0,00	0,00	0,00
cpmp07	787	0,00	0,12	0,00	0,35	0,00	2,28	0,00
cpmp08	820	0,00	0,86	0,00	4,65	0,00	0,00	0,12
cpmp09	715	0,00	0,12	0,00	0,24	0,00	0,00	0,00
cpmp10	829	0,00	0,13	0,00	0,89	0,00	0,00	0,00
cpmp11	1006	0,00	0,63	0,00	1,83	0,00	0,00	0,29
cpmp12	966	0,00	0,20	0,00	1,23	0,00	0,00	0,20
cpmp13	1026	0,00	0,24	0,00	0,43	0,00	0,00	0,00
cpmp14	982	0,00	1,80	0,00	5,15	0,30	0,00	0,30
cpmp15	1091	0,00	3,09	0,00	6,57	0,00	0,00	0,36
cpmp16	954	0,00	0,23	0,00	0,80	0,00	0,00	0,31
cpmp17	1034	0,00	0,32	0,00	2,16	0,48	0,29	0,58
cpmp18	1043	0,00	0,22	0,00	2,26	0,19	0,19	0,19
cpmp19	1031	0,00	0,32	0,00	2,42	0,00	0,09	0,29
cpmp20	1005	0,00	4,61	0,00	64,49	0,00	1,39	0,00
Médias		**0,00**	**0,68**	**0,00**	**4,71**	**0,05**	**0,36**	**0,28**

Fonte: Elaborado pelo autor (2017)

Os mesmos critérios de parada utilizados nas instâncias *cpmp* foram empregados nas *sjc*. Os parâmetros do CRO que permaneceram constantes em todas as execuções são: *KELossRate = 0,8, MoleColl = 0,1, InitialKE = 1.000.000, buffer = 0, MinMol = 2, MaxMol = 100, Minλ-InterchangeIterations = 1*, $\kappa 0 = 1$ *e* $\Delta\kappa = 1$. Uma otimização *1-opt* foi efetuada em todas as colisões ineficazes, com exceção da instância *sjc4a*, na qual foi utilizada uma *2-opt*, somente para as colisões intermoleculares. A Tabela 8 mostra os resultados computacionais para o conjunto de instâncias *sjc*. O CRO atingiu o valor ótimo em 4 de 6 instâncias, com um *gap* e tempo de execução médios de 0,49% e 88.17s respectivamente, dentre todos os experimentos.

A Tabela 9 mostra uma comparação do CRO com outros métodos heurísticos, exatos e híbridos. A média dos melhores *gaps* obtidos pelo CRO foi de 0,124% com um tempo médio de execução de 54,30s. Em comparação com os resultados relatados por Stefanello e colaboradores (2015), na resolução do modelo completo via CPLEX vs. 12.3, o CRO foi um pouco mais rápido (3.40s). No entanto, o CRO foi 38,55 mais lento que o IRMA (na configuração $\alpha = 2,4$), mas muito mais rápido do que os outros métodos no grupo. Em média, foi 381,03s mais rápido que o *Fen-Cplex*, o terceiro método mais rápido, e 3055,81s mais rápido que o *VNS*, o mais lento do grupo. Em relação à otimalidade, a heurística *SS* a conseguiu em apenas 3 instâncias, sendo a pior no grupo, enquanto os outros métodos atingiram valores ótimos em 4 ou mais instâncias, igualando ou superando o CRO. Enquanto que a média dos melhores *gaps* alcançados pelo CRO possa, na opinião do autor, ser considerada satisfatória, a mesma foi maior do que a dos outros métodos, especialmente dos que usaram um otimizador MIP (CPLEX).

Apesar de não ter sido o método mais rápido ou mais preciso nesta comparação, o autor acredita que a presente implementação do CRO para CPMP possa ter uma vantagem competitiva em aplicações que devam ser executadas em plataformas de hardware mais modestas, como aquelas usadas em sistemas embarcados ou dispositivos móveis. A razão é que é que o CRO requer um tamanho de memória pequeno (normalmente menor que 64 MB nos testes efetuados) e é executado em uma única CPU, ou núcleo, enquanto a maioria das soluções híbridas que empregam otimizadores MIP, como o IBM CPLEX, exigem grandes quantidades de memória RAM (geralmente acima de 4GB) e CPUs *multicore* para atingir um bom desempenho. O alto custo de licenciamento de um otimizador MIP também pode ser um fator limitante para tais soluções híbridas, dependendo da aplicação. Além disso, os *gaps* e tempos de processamento que podem ser obtidos com o CRO podem ser aceitáveis em muitos cenários e aplicações.

4.2.3.3 Instâncias adaptadas da TSP-LIB

A última das bases de dados utilizada em comparações de *gaps* e tempos de execução foi proposta por Stefanello e colaboradores (2015) para avaliação de seu método mateurístico IRMA e é composta por 3 conjuntos de dados de 5 instâncias cada, todos adaptados de *TSP-LIB*, com 318 a 724 vértices e 5 a 200 medianas.

Da mesma forma que no trabalho de Stefanello e colaboradores (2015), todos os problemas foram resolvidos 10 vezes utilizando o CRO e os resultados, então, foram comparados com os obtidos pelas fases 2 e 3 do IRMA.

Como a maioria desses problemas não pode ser resolvida à otimalidade por otimizadores MIP num tempo computacional aceitável, foram relatados na Tabela 10 os valores da função objetivo para a melhor solução viável (MSV) encontrada, independentemente de terem sido alcançados por métodos exatos, CRO ou IRMA. Sempre que valores ótimos estiverem disponíveis, os mesmos estarão marcados em negrito.

A Tabela 10 mostra os *gaps* médios e os tempos médios de execução para as fases 2 e 3 do IRMA, bem como para a CRO. Mostra também, apenas para CRO, a porcentagem de vezes que a MSV é atingida, o melhor *gap*, o desvio padrão, o número médio de iterações, o número médio de decomposições, colisões com a parede, sínteses e colisões intermoleculares. Por fim, estão mostrados na Tabela 10 os valores definidos para *MaxIterations, MaxIterationsWithoutImprovement* e *PopSize*. Todos os outros parâmetros do CRO permanecem constantes para todas as instâncias: *PopSize = 10; KELossRate = 0,8, MoleColl = 0,1, InitialKE = 1.000.000,* α *= 10,* β *= 100.000, buffer = 0, MinMol = 2, MaxMol = 100 e Min*λ*-InterchangeIterations = 1.* $\kappa 0$ *e* $\Delta\kappa$ foram definidos como 1 para todas as instâncias, exceto para *u724_125* e *u724_200*, que tiveram ambos configurados para 0,1. Uma otimização *1-opt* foi usada em todas as colisões ineficazes.

Tabela 8 – Resultados computacionais do CRO para as instâncias *sjc*

INSTÂNCIA				**CRO**														
ID	***n***	***p***	**Z**	**MSV (%)**	***Gap Médio* (%)**	***Melhor gap* (%)**	**Desvio Padrão**	**Tempo Médio Exec. (s)**	**Média Iter.**	**Med. Decomp.**	**Média Col. Parede**	**Média Sint.**	**Média Col. Inter.**	***Max Iter.***	***Max. Iter. w/o Improv***	***Pop Size***	***α***	***β***
sjc1	100	10	17288,99	36,67	0,49	0.00	72,23	3,08	221	3	193	0	23	300	150	10	10	50000
sjc2	200	15	33270,94	26,67	0,32	0.00	89,16	1,83	55	0	48	0	6	100	50	10	10	50000
sjc3a	300	25	45335,16	3,33	0,27	0.00	50,44	250,95	4136	473	3212	300	150	5000	2500	50	2	500000
sjc3b	300	30	40635,90	3,33	0,26	0.00	49,08	49,89	791	94	611	62	23	1000	500	10	2	500000
sjc4a	402	30	61925,51	0,00	0,82	0.37	84,78	211,64	854	15	748	9	80	1000	500	10	10	50000
sjc4b	402	40	52458,00	0,00	0,76	0.37	99,18	11,60	66	0	58	0	7	100	50	10	10	50000
Médias					**0,49**	**0,12**		**88,17**										

Fonte: Elaborado pelo autor (2017)

Tabela 9 – Comparação do CRO com outras heurísticas para as instâncias *sjc*

Instância		**CRO**		**CPLEX**		**IRMA (α = 2.4)**		**SS**		**VNS**		**CS**		**Fen-Cplex**	
ID	**Z**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**	**Melhor *gap* (%)**	**Tempo Exec. (s)**
sjc1	17288,99	0,000	1,17	0,000	4,89	0,000	1,90	0,000	60,00	0,000	50,50	0,000	22,72	0,000	37,60
sjc2	33270,94	0,000	0,53	0,000	11,46	0,000	3,25	0,068	600,00	0,000	44,08	0,000	112,81	0,000	127,90
sjc3a	45335,16	0,000	198,93	0,000	62,20	0,000	23,96	0,006	2307,00	0,000	8580,30	0,000	940,75	0,000	495,10
sjc3b	40635,90	0,000	1,30	0,000	16,14	0,000	2,47	0,000	2308,00	0,000	2292,86	0,000	1887,97	0,000	72,20
sjc4a	61925,51	0,374	116,79	0,000	215,60	0,000	56,67	0,000	6109,00	0,000	4221,47	0,005	2885,11	0,000	1209,50
sjc4b	52458,00	0,369	7,08	0,000	35,90	0,000	6,26	0,140	6106,00	0,023	3471,44	0,140	7626,33	0,000	669,70
Médias		**0,124**	**54,30**	**0,000**	**57,70**	**0,000**	**15,75**	**0,036**	**2915,00**	**0,004**	**3110,11**	**0,024**	**2245,95**	**0,000**	**435,33**

Fonte: Elaborado pelo autor (2017)

Os resultados dos experimentos efetuados mostraram que o CRO apresentou um desempenho melhor ao resolver instâncias em que *p* (número de medianas) é pequeno. Mais especificamente, variando de 5 a 15 medianas. Dentro deste intervalo, o CRO foi capaz de atingir a MSV em, pelo menos, 30 por cento das execuções, e com *gaps* médios abaixo de 0,08%. À medida em que *p* aumenta, os resultados se tornam menos precisos e os tempos de execução aumentam significativamente. Este comportamento pode ser explicado pela natureza do mecanismo *λ-interchange* utilizado, no qual cada cliente de um determinado agrupamento é sistematicamente reconectado a todas as estações selecionadas nas proximidades da estação à qual o mesmo está correntemente conectado, procurando por possíveis melhorias no valor da função objetivo. No mecanismo *λ-interchange* tradicional, para um dado λ, existem *p (p - 1) / 2* pares diferentes de agrupamentos a serem examinados, ou seja, o número de iterações cresce com o quadrado de *p* (OSMAN; CHRISTOFIDES, 1994). Para mitigar tal problema, o Conjunto de Listas de Proximidade tenta reduzir o número de iterações efetuadas pelo mecanismo, limitando indiretamente a quantidade de pares de agrupamentos a serem avaliados, conforme explicado no item 3.6.4. Ainda assim é inevitável que os tempos de execução aumentem demasiadamente para valores muito altos de *p*.

Para as instâncias testadas, o pior *gap* médio e tempo de execução foram, respectivamente, 3,56% e 3621,24, para a instância *ali_535_150*. Embora o melhor *gap* e tempo médio de execução tenham sido obtidos com a instância *ali_535_005*, a MSV foi obtida durante a fase construtiva do CRO, não necessitando, portanto, de quaisquer iterações do *loop* principal do mesmo. Desta forma, os melhores resultados, que necessitaram de iterações, do CRO foram obtidos com a instância *ali_318_005*, com *gap* de 0% e tempo de execução médio de 9.76s.

Em comparação com a Fase 2 do IRMA, a versão mais rápida e menos precisa do IRMA, o CRO atingiu *gaps* iguais ou melhores em 6 das 15 instâncias e foi mais rápido em 5 delas. Em média, O CRO foi mais lento que IRMA Fase 2, com média de 900,71s versus 248,21s. O *gap* médio de todas as instâncias foi de 1,17%, portanto, 0,45% maior do que a Fase 2 do IRMA, cujo *gap* é de 0,72%.

A Fase 3 do IRMA é a mais precisa e inclui o processamento realizado na Fase 2. Como nem todas as instâncias testadas por Stefanello e colaboradores (2015) exigiram a Fase 3 para obter resultados satisfatórios, a comparação com o CRO ficou restrita às instâncias reportadas pelo autor. O CRO foi mais rápido em 5 de 15 instâncias, mas obteve *gaps* iguais ou piores em todas as instâncias testadas. Em média, a diferença percentual do *gap* médio entre CRO e IRMA

Fase 3 foi de 1,14%. Como na comparação com a Fase 2, o CRO foi mais lento que a Fase 3 do IRMA, com média de 900,71s versus 471,30s.

Da mesma forma que nas instâncias *sjc*, apesar de ter apresentado *gaps* e tempos de execução médios maiores que os obtidos por pelas Fases 2 e 3 IRMA, o autor acredita que a presente implementação do CRO para CPMP possa ter uma vantagem competitiva em aplicações que devam ser executadas em plataformas de hardware mais modestas devido ao baixo consumo de *CPU* e memória.

Tabela 10 – Resultados e comparação do CRO com IRMA, para TSP-LIB

Instância				**IRMA Fase 2**		**IRMA Fase 3**		**CRO**											
ID	***n***	***p***	**MSV**	***Gap* Med. (%)**	**Tempo Médio Exec.**	***Gap* Med (%)**	**Tempo Médio Exec.**	***Gap Med* (%)**	**Tempo Médio Exec,**	**MSV (%)**	**Melhor *gap* (%)**	**Desv. Padrão**	**Med. Iter.**	**Med. Dec.**	**Média Col. Parede**	**Med. Sint.**	**Med Col. Inter**	**Max Iter.**	**Max Iter. w/o Impr.**
lin318_005	318	5	**180281,21**	0,00	9,15	-	-	0,00	9,76	60	0,00	0,58	135	3	117	0	14	200	100
lin318_015	318	15	**88901,56**	0,00	26,35	-	-	0,08	23,88	60	0,00	107,09	268	5	234	2	26	300	150
lin318_040	318	40	**47988,38**	1,01	222,41	0,14	319,46	0,65	97,76	0	0,39	143,22	483	4	424	8	44	500	250
lin318_070	318	70	**32198,64**	0,01	127,45	-	-	1,02	728,49	0	0,42	110,87	3477	35	3063	42	189	4000	2000
lin318_100	318	100	**22942,69**	2,23	222,65	0,00	364,87	0,97	2075,61	0	0,50	54,68	3518	28	3109	35	167	4000	2000
ali535_005	535	5	**9956,77**	0,00	7,08	0,00	45,42	0,00	2,17	100	0,00	0,00	0	0	0	0	0	4000	2000
ali535_025	535	25	**3695,15**	0,24	311,36	0,00	544,26	0,07	348,36	0	0,04	1,18	883	12	777	19	47	1000	500
ali535_050	535	50	2461,41	1,69	377,28	0,00	726,30	1,58	842,99	0	0,77	14,50	983	7	867	14	55	1000	500
ali535_100	535	100	1438,42	2,61	362,75	0,02	637,64	3,17	1341,98	0	2,08	11,08	969	4	863	12	43	1000	500
ali535_150	535	150	1032,28	2,54	366,54	0,00	761,31	3,56	3621,14	0	2,40	9,78	953	3	845	11	47	1000	500
u724_010	724	10	181782,96	0,00	6,64	0,00	59,65	0,03	354,99	30	0,00	88,99	381	7	330	4	38	500	250
u724_030	724	30	**95034,01**	0,01	158,05	0,00	300,72	0,14	152,50	0	0,08	49,40	408	9	354	7	37	500	250
u724_075	724	75	54735,05	0,08	507,56	0,00	546,39	0,96	439,98	0	0,62	99,18	486	3	434	9	36	500	250
u724_125	724	125	38976,76	0,28	509,01	0,02	643,31	2,52	761,39	0	1,82	185,7	981	4	871	12	57	1000	500
u724_200	724	200	28079,97	0,10	508,81	0,11	706,29	2,75	2710,03	0	2,17	140,68	976	3	861	11	57	1000	500
Médias				**0,72**	**248,21**	**0,02**	**471,30**	**1,17**	**900,74**		**0,75**								

Fonte: Elaborado pelo autor (2017)

4.3 RESULTADOS PARA O MODELO FCLP BAOUCHE

4.3.1 Avaliação da adaptação do CRO para o modelo FCLP Baouche

Para a avaliação da adaptação da metaheurística CRO para o modelo de localização de estações de carregamento de EVs (FCLP Baouche), proposto por Baouche e colaboradores (2014), o seguinte experimento foi realizado utilizando a base *vix*, composta de 5 instâncias com 50 a 100 vértices, conforme descrito na etapa 3.7, da seção Metodologia.

4.3.1.1 Descrição dos experimentos

Neste experimento, os resultados da resolução da base de dados *vix* pelo CRO para o modelo FCLP Baouche, foi comparado, em termos de *gaps* e tempos de execução, com os obtidos pela resolução, de forma exata, pelo otimizador CPLEX, utilizando-se do mesmo hardware.

Cada uma das instâncias *vix* foi resolvida, de forma exata, utilizando o otimizador IBM CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) vs. 12.6. A seguir, na mesma plataforma computacional, cada instância foi resolvida 30 vezes pelo aplicativo CRO, utilizando-se a mesma configuração para todos os problemas.

A configuração dos parâmetros operacionais do CRO utilizada na resolução dos problemas foi obtida empiricamente, na forma de tentativa e erro, sendo a seguinte: *PopSize = 10, KELossRate = 0,8, MoleColl = 0,1, InitialKE = 1.000.000.000, α = 10, β = 100.000.000, buffer = 0, MinMol = 2, MaxMol = 100, Minλ-InterchangeIterations = 1, κ0 = 1 e $\Delta\kappa$ = 1*. Como critério de parada, limitou-se o número máximo de iterações (*MaxIterations*) em 10.000 e o número máximo de iterações realizadas sem melhorias (*MaxIterationsWithoutImprovement*) em 5.000 iterações. Adicionalmente, o programa do CRO foi configurado para terminar caso o valor ótimo ou a melhor solução viável (*MSV*) para a instância fosse atingido.

Devido a eletrificação da frota brasileira de automóveis ser praticamente inexistente, valores constantes foram atribuídos à alguns dos dados de entrada do modelo FCLP Baouche, como se segue:

- $Di = 24, \forall\, i \in I$
- $r = 5000$
- $n_i^{(ev)} = 4, \ \forall\, i \in I$

- $f_j = 600.000, \forall\, j \in J$
- $c_j = 4.800, \forall\, j \in J$
- $\propto = 540$

O item 3.7.3, detalha como os valores constantes acima foram obtidos.

4.3.1.2 Resultados

Todas as instâncias foram resolvidas à otimalidade via CPLEX, exceto *vix250*. Diversas tentativas foram feitas para resolver a mesma, porém o CPLEX sempre abortava a execução, por falta de memória de trabalho, após cerca de duas horas e meia de processamento, tornando-se instável e obrigando o equipamento a ser reinicializado. Desta forma, limitou-se o tempo máximo para resolução do modelo em 7200s, registrando-se o valor da função objetivo obtido como sendo o da melhor solução viável (MSV) encontrada.

A Tabela 11 apresenta os resultados da resolução das instâncias *vix* pelo CPLEX. A coluna ID representa a identificação da instância. As colunas *n* e *p* mostram o número de vértices e o número de medianas selecionado pelo CPLEX, respectivamente. A coluna MSV exibe a melhor solução viável (MSV) encontrada, sendo que os valores em negrito representam valores ótimos. A mesma tabela exibe os resultados obtidos pelo CRO. São mostrados o número mínimo e máximo de medianas selecionadas, correspondendo às estações de carregamento de EVs selecionadas, o *gap* e o tempo de execução médios, o percentual de atingimento do MSV, o desvio padrão, o número médio de iterações, decomposições, colisões com a parede, sínteses e colisões intermoleculares.

A seguir encontram-se os resultados obtidos pelo CRO, por instância:

vix050: A otimalidade foi atingida em todas as execuções de teste. Sendo uma instância de fácil resolução, tanto CRO como o CPLEX a resolveram em tempos abaixo de 1s, sendo o que o CRO foi 0,13s mais lento.

vix0100: A otimalidade foi atingida em todas as execuções de teste. O CRO foi 4,35s mais lento que o CPLEX.

vix0150: A otimalidade foi atingida em todas as execuções de teste. O CRO foi 17.01s mais lento que o CPLEX.

vix0200: O CRO não atingiu a otimalidade em nenhuma das execuções de teste. Porém o menor *gap* foi de apenas 0,05%, sendo a média de 0,45%. O CRO foi 197.96s mais lento que o CPLEX.

vix0250: O CRO obteve nítida vantagem em relação ao CPLEX, em termos de tempo de execução, na resolução desta instância. Conforme explicado anteriormente, o CPLEX teve seu tempo de execução limitado a 2 horas, uma vez que, após consumir toda a memória disponível no sistema (8 GB) o mesmo abortava a execução. Assim, o resultado obtido não foi ótimo, sendo que o *gap* reportado foi de 0,21%, ao término do tempo de processamento estipulado (7200s). O CRO não atingiu a otimalidade em nenhuma das execuções de teste. O menor *gap*, em relação a MSV, foi de 0,72% e a média, 0,98%. O tempo médio de execução ficou um pouco acima de 6 minutos (385,89s).

Em média, o CRO foi 1323,43s mais rápido que o CPLEX, em grande parte devido à instância *vix250*. Os *gaps* médios apresentados pelo CRO estiveram sempre abaixo de 1% em relação aos valores ótimos ou melhores soluções viáveis conhecidas.

Com a enorme evolução na velocidade de processamento dos processadores disponíveis para PCs nos últimos 15 anos e, notadamente, com o surgimento de *CPUs multicore* à preço acessível no final da década passada, a resolução por otimizadores MIP, como o CPLEX, de instâncias relativamente grandes de problemas de programação linear e inteira se tornou uma alternativa viável ao uso de metaheurísticas puras. No caso específico do CPMP, isso pode ser comprovado pelo número de trabalhos científicos publicados que fazem uso de soluções híbridas, onde uma metaheurística é utilizada na redução de variáveis do modelo ou construção de uma solução inicial, ficando a cargo de um otimizador MIP a resolução à otimalidade da solução entregue pela metaheurística. Várias soluções híbridas desenvolvidas recentemente são mostradas no item 2.1.2.2, da revisão de literatura, confirmando essa tendência. Resultados publicados por Boccia e colaboradores (2008) e Stefanello e colaboradores (2015) comprovaram a eficácia de tais soluções híbridas mesmo na resolução de instâncias grandes, com mais de 1000 vértices, onde a resolução direta por otimizadores MIPs é inviável.

Diante do exposto e levando-se em conta que a presente implementação do CRO é uma adaptação de uma metaheurística pura, portanto, sem o auxílio de otimizadores MIP, e com baixo consumo de CPU e memória RAM, o autor considera que os resultados obtidos foram satisfatórios.

4.3.2 Avaliação do modelo FCLP Baouche na localização de estações

Para a avaliação do comportamento do modelo proposto por Baouche e colaboradores (2014), na localização de estações de EVs na região da Grande Vitória, ES, e utilizando a metaheurística CRO, os seguintes experimentos foram realizados utilizando a instância *vix100*, composta de 100 vértices, representando tanto locais candidatos a receber estações de carregamento de EVs como centros de consumo contendo EVs necessitando de recarga.

4.3.2.1 Descrição dos Experimentos

Utilizando a mesma plataforma computacional e parâmetros de configuração do CRO, do item 4.3.1.1, a instância *vix100* foi resolvida pelo aplicativo CRO, utilizando-se o modelo FCLP Baouche (BAOUCHE et al., 2014), para diferentes tamanhos de frota circulante de EVs necessitando de recarga, de forma a simular diferentes taxas de penetração de EVs na Grande Vitória. O objetivo desta análise de sensibilidade foi o de determinar o comportamento do modelo FCLP Baouche na localização de estações à medida que o número de EVs em circulação crescia.

Nos testes, a taxa de penetração de EVs foi variada entre 0,01% e 0,1% da frota atual de automóveis estimada da Grande Vitória (1 milhão), em intervalos de 0,01%. Para tanto, o número de veículos necessitando de recarga, $n_i^{(ev)}$, $\forall\, i \in I$, foi variado entre 1 e 10, em incrementos de 1 unidade, o que corresponde a tamanhos de frota entre 100 e 1000 EVs, já que a instância *vix100* possui 100 centros de consumo (vértices). Os demais valores relativos ao conjunto de dados de entrada permaneceram os mesmos do item 4.3.1.1, ou seja:

- $Di = 24, \forall\, i \in I$
- $r = 5000$
- $f_j = 600.000, \forall\, j \in J$
- $c_j = 4.800, \forall\, j \in J$
- $\propto = 540$

Para cada valor distinto de $n^{(ev)}$, o problema resultante foi resolvido 10 vezes pelo CRO, sendo aproveitado o melhor resultado, ou seja, aquele com o menor valor de função objetivo.

4.3.2.2 Resultados

A Tabela 12 mostra os resultados da análise de sensibilidade efetuada para diferentes taxas de penetração de EVs na frota de automóveis da Grande Vitória. Para cada tamanho de frota de EVs são mostrados o número de estações selecionadas, o melhor valor da função objetivo alcançado e respectivo tempo de execução, além dos índices $j, \forall\, j \in J$, para os quais $x_j = 1$, correspondente às estações selecionadas no modelo FCLP Baouche (1) .

Como esperado, o número de estações aumenta com a taxa de penetração, exceto para as configurações de 600 e 700 EVs, onde as mesmas estações são selecionadas. As Figuras 26, 27 e 28 exibem, no mapa da Grande Vitória, os locais onde as estações de carregamento foram localizadas, para frotas de 100, 500 e 1000 EVs, respectivamente. Os círculos representam os centros de consumo, onde estão concentradas as demandas de carregamento, e os marcadores numerados em forma de gota, as estações de carregamento selecionadas.

A partir de uma análise visual dos mapas de localização de estações, pode-se concluir que, em quase todos os experimentos, estações de carregamento foram localizadas próximas aos centros de consumo. Apenas no mapa com apenas 5 estações abertas (Figura 26), alguns centros de consumo, nos munícipios de Fundão e Viana ficaram distantes das estações mais próximas, o que poderia inviabilizar a operação de EVs em tais locais. Este problema poderia ser mitigado, atribuindo-se a locais mais distantes da capital, estações de carregamento com capacidades mais modestas e menor custo. No entanto, à medida que o número de EVs circulando nos locais mais distantes aumenta, estações passam a ser localizadas nos locais mais extremos, como mostram as Figuras 27 e 28, incluindo pontos ao longo do litoral e na rodovia BR-101, próximo a Guarapari (Figura 28).

Tabela 11 – Resultados computacionais do CRO e CPLEX para as instâncias *vix*

Instância		**CPLEX 12.6**			**CRO**											
ID	***n***	***p***	**MSV**	**Tempo Exec. (s)**	**Min. Med.**	**Max. Med.**	***Gap* Médio (%)**	**Tempo Médio Exec. (s)**	**MSV (%)**	**Melhor *gap* (%)**	**Desvio Padrão**	**Med. Iter. (%)**	**Med. Decomp. (%)**	**Med. Col. Parede (%)**	**Med. Sint. (%)**	**Med. Col. Inter. (%)**
vix050	50	9	**10670960**	0,14	9	9	0,00	0,27	100	0,00	0,00	32	1	27	0	4
vix100	100	14	**16617412**	1,06	14	14	0,00	5,41	100	0,00	0,00	668	29	566	22	50
vix150	150	14	**21749634**	4,95	14	14	0,00	21,96	100	0,00	0,00	1827	80	1548	68	128
vix200	200	16	**27039387**	50,27	16	19	0,45	248,23	0	0,05	70529,47	8200	315	6996	304	580
vix250	250	21	34605868	7222,47	21	26	0,98	385,89	0	0,72	135124,68	8153	268	6995	270	614
Médias				**1455,78**			**0,29**	**132,35**		**0,15**						

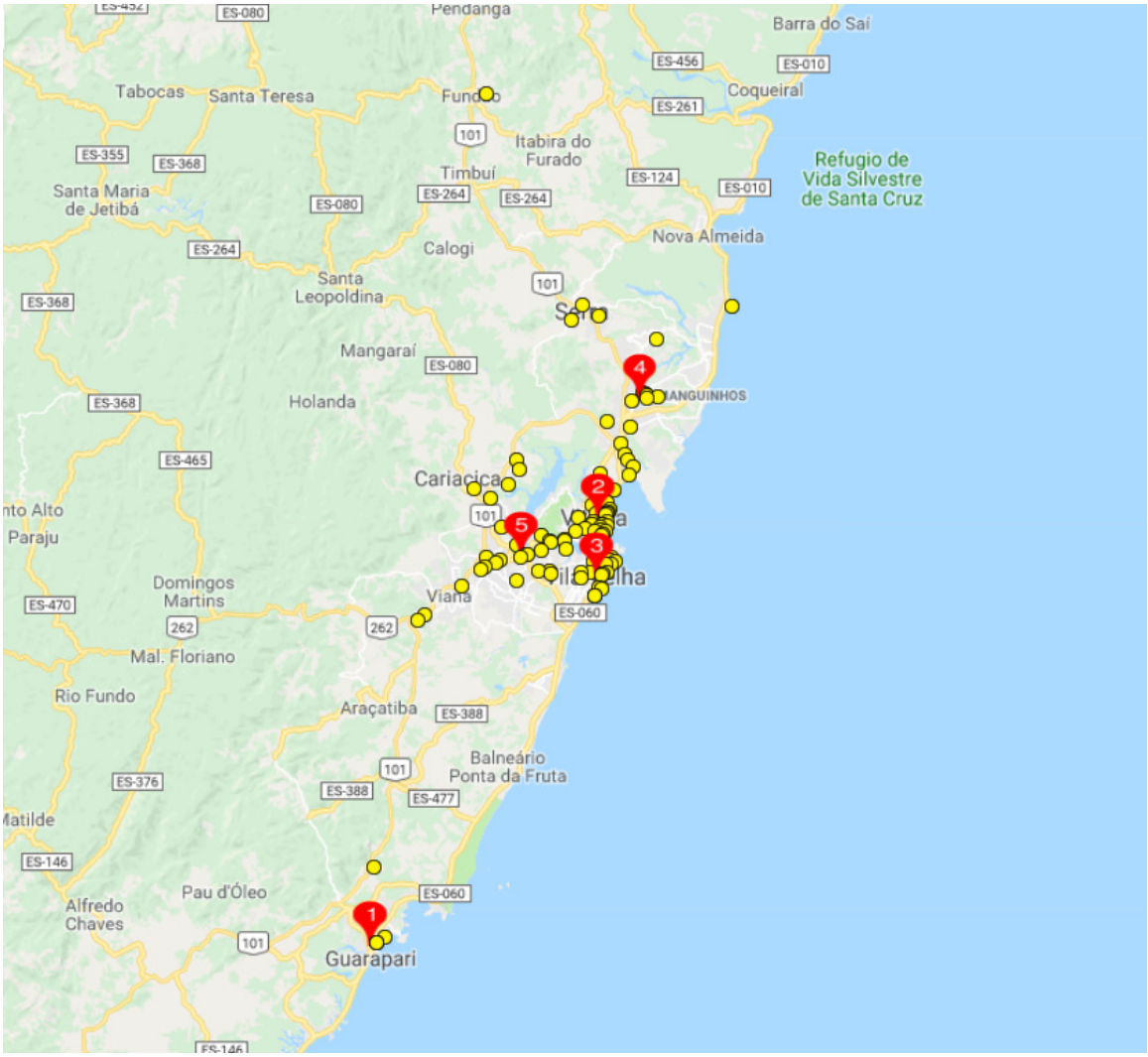
Fonte: Elaborado pelo autor (2017)

Tabela 12 – Estações selecionadas em função da frota de EVs, para *vix100*

Tam. Frota EVs	Num. Estações Selecionadas	Melhor Objetivo	Tempo Exec. para melhor objetivo(s)	Estações de Carregamento de EVs Selecionadas																						
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
100	5	7575639	0,374	5	55	71	74	95																	
200	8	11289168	0,283	5	22	33	55	71	74	76	95														
300	12	14267546	0,361	2	5	10	20	21	22	24	25	30	33	71	74										
400	14	16617412	2,702	5	10	20	21	22	25	28	30	33	35	41	46	71	74								
500	15	18960773	36,066	2	5	10	16	20	21	22	25	26	27	28	30	41	46	74							
600	17	22051351	75,433	2	7	10	16	20	21	22	25	26	28	30	38	41	46	50	74	89					
700	17	24936058	109,974	2	7	10	16	20	21	22	25	26	28	30	38	41	46	50	74	89					
800	18	28185688	122,428	6	7	16	20	21	22	26	28	30	35	38	41	46	48	50	58	74	89				
900	21	31907376	105,743	6	7	16	20	21	22	23	26	28	30	38	41	46	47	48	50	58	65	74	76	89	
1000	22	43191941	242,735	6	7	12	16	20	21	23	30	33	38	40	41	46	47	48	50	58	59	65	74	76	89

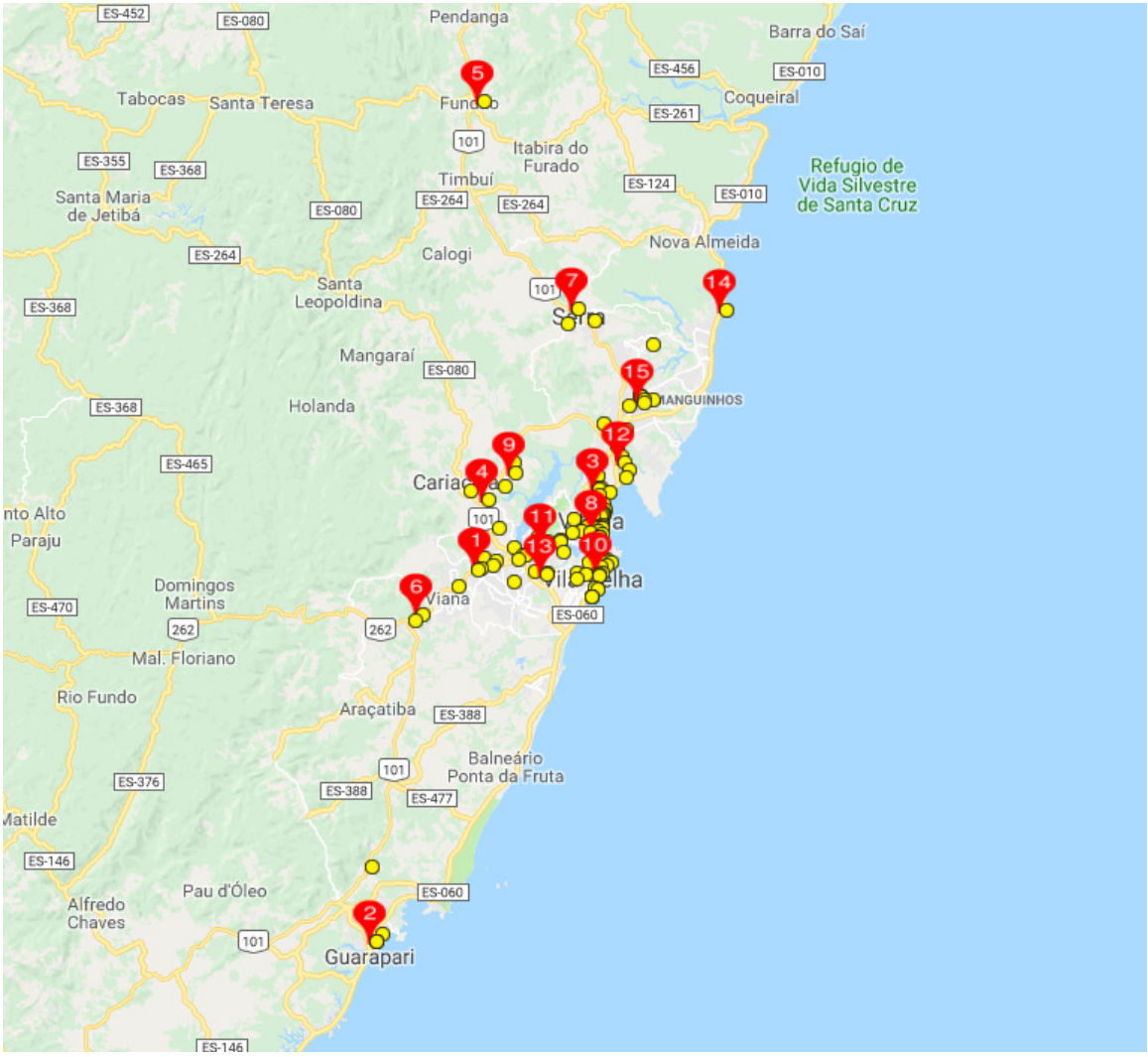
Fonte: Elaborado pelo autor (2017)

Figura 26 – Localização de estações de carregamento de EVs, para frota de 100 veículos



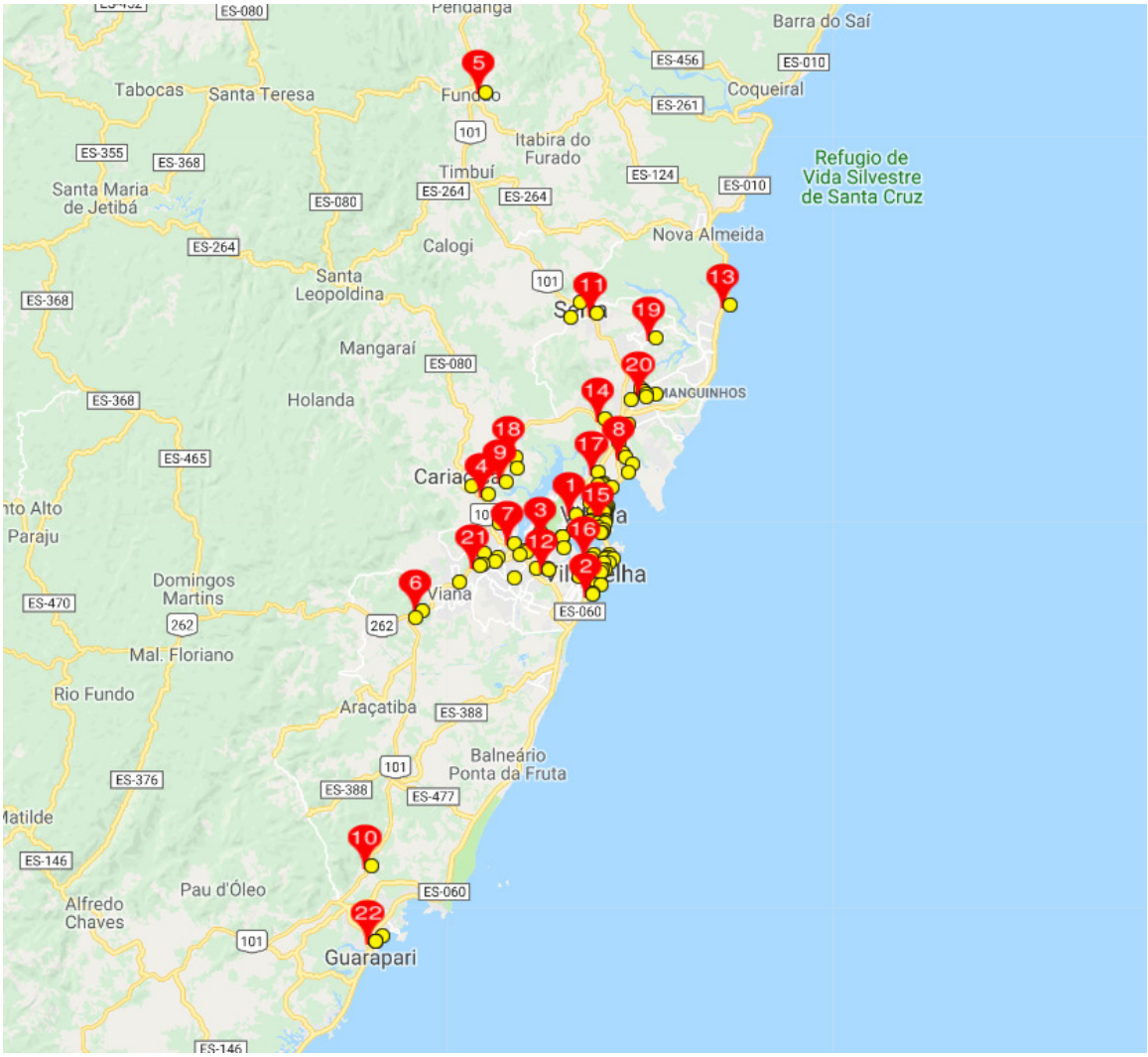
Fonte: Elaborado pelo autor (2017)

Figura 27 – Localização de estações de carregamento de EVs, para frota de 500 veículos



Fonte: Elaborado pelo autor (2017)

Figura 28 – Localização de estações de carregamento de EVs, para frota de 1000 veículos



Fonte: Elaborado pelo autor (2017)

5 CONCLUSÃO E TRABALHOS FUTUROS

Nesse trabalho, um problema matemático de otimização para localização de estações de carregamento de EVs foi utilizado para situar, de forma eficiente, estações na Grande Vitória, ES. O problema escolhido da literatura destina-se a minimizar o custo de implantação das estações de carregamento, bem como os custos de deslocamento dos usuários de EVs às mesmas.

Uma base de dados, contendo um conjunto inicial de locais candidatos a receberem estações de carregamento de EVs e centros de consumo, foi criada para a região da Grande Vitória, a partir de dados obtidos de sistemas de mapeamento e navegação disponíveis na Internet. A demanda de recarga foi representada como agrupamentos de EVs localizados nos centros de consumo selecionados. Utilizaram-se também distâncias reais de percurso entre tais agrupamentos e estações de carregamento, disponibilizadas pelos mesmos sistemas de navegação. Tais dados serviram como base para a criação de cinco instâncias com tamanhos variando de 50 a 250 vértices (estações candidatas/centros de consumo).

Versões específicas da metaheurística *Chemical Reaction Optimization* (CRO) foram implementadas para a resolução do problema de localização escolhido, além de problemas de localização semelhantes (p-mediana capacitada e não-capacitada), utilizados conjuntamente com bases de dados de referência da literatura, para avaliar a qualidade das implementações. A seguir estão relacionadas as principais atividades realizadas durantes as fases desenvolvimento, teste e calibração das diferentes implementações do CRO:

- Implementação, em linguagem C#, do *framework* original do CRO (LAM; LI, 2012).
- Adaptação e implementação da classe Molécula (LAM; LI, 2012).
- Criação e implementação da estrutura de dados denominada Conjunto de Listas de Proximidade (CLP) e algoritmos utilizados para sua construção e atualização.
- Criação e implementação de algoritmo para construção de novas soluções.
- Implementação do algoritmo *Fast Interchange*, utilizado em colisões ineficazes no modelo de p-mediana não-capacitada e na construção de novas soluções (WHITAKER, 1983).

- Implementação do algoritmo *Neighborhood Search*, utilizado na construção de novas soluções (MARANZANA, 1964).

- Implementação do algoritmo *Half-Total Change* utilizado em decomposições moleculares (LAM; LI, 2012).

- Implementação do algoritmo *Distance Preserving Crossover* utilizado em sínteses moleculares (MERZ; FREISLEBEN, 1997).

- Implementação e adaptação do algoritmo de busca local *λ-interchange*, proposto Osman e Christofides (1994), para operar com a estrutura de dados Conjunto de Listas de Proximidade (CLP). O algoritmo foi utilizado em colisões ineficazes no modelo de p-mediana capacitada e FCLP Baouche

Algumas das bases de dados utilizadas também foram resolvidas pelo otimizador IBM CPLEX, a fim de poder-se avaliar a qualidade da adaptação do CRO. Para tanto, os modelos associados a tais bases tiveram que ser codificados na linguagem de programação orientada a otimização OPL, adotada pelo CPLEX.

Como resultado deste esforço, uma ferramenta de suporte à decisão, na forma de um aplicativo desenvolvido em linguagem de programação C#, foi criada, que poderá auxiliar na futura implantação de estações de carregamento em áreas metropolitanas.

Além da ferramenta de suporte à decisão, o presente trabalho produziu os seguintes artigos: "Solução do problema da p-mediana para localização de estações de serviço utilizando metaheurística chemical reaction optimization" (SILVA; MESTRIA, 2017), publicado no Simpósio Brasileiro de Pesquisa Operacional (SBPO) 2017 e "Chemical Reaction Optimization metaheuristic for locating service stations through the capacitated p-median problem" (SILVA; MESTRIA, 2018), submetido ao periódico Revista Pesquisa Operacional, e que encontra-se atualmente em revisão.

As implementações do CRO para os modelos de p-mediana capacitada e não-capacitada, desenvolvidas para este trabalho, parecem ser competitivas em relação aos métodos avaliados, embora não se possa afirmar que o CRO seja o melhor método para a solução de tais problemas. Ainda assim, é possível que encontrem aplicabilidade em diversos cenários e aplicações, pois não requerem otimizadores MIP (potencialmente complexos e caros), são executadas em um

único processador com baixo uso de memória, sendo capazes de fornecer resultados com *gaps* e tempos de execução que podem ser aceitáveis em muitas situações da vida real.

Com respeito a adaptação do CRO para solução do problema de localização de Baouche e colaboradores (2014), pode-se concluir, a partir dos resultados obtidos, que a metaheurística também é aplicável a tal problema, uma vez que foi capaz de chegar a soluções de boa qualidade com baixo tempo computacional.

No que tange a utilização do modelo de Baouche e colaboradores (2014), na localização de estações de carregamento de EVs na Grande Vitória, o mesmo provou ser adequado para a tarefa, concentrando estações de carregamento próximas às áreas de maior densidade urbana, sem, no entanto, deixar de atender a locais mais distantes, à medida em que a penetração de EVs aumentava.

Existem várias possibilidades de pesquisas e trabalhos futuros. Dentre elas está o desenvolvimento de uma versão do CRO que utilize processamento paralelo, fazendo melhor uso do hardware moderno, caracterizado por CPUs *multicore*. Tal versão poderá ser capaz de lidar, de forma eficiente, com instâncias grandes, de alguns milhares de vértices. Além disso, outros operadores mais eficientes poderão ser desenvolvidos para as colisões moleculares, melhorando assim a qualidade dos resultados.

Quanto ao modelo de Baouche e colaboradores (2014) , o mesmo poderia ser estendido, a fim de incluir uma restrição que limitasse a distância máxima entre um agrupamento de EVs e a estação de carregamento designada, impedindo que centros de consumo mais distantes deixassem de ser atendidos por estações de carregamento, por estarem localizados além de uma distância técnica ou economicamente viável para abastecimento, das mesmas.

REFERÊNCIAS

AGENBROAD, Josh; HOLLAND, Ben. **RMI: What's the true cost of EV charging stations?** Disponível em: <https://www.greenbiz.com/blog/2014/05/07/rmi-whats-true-cost-ev-charging-stations>. Acesso em: 18 fev. 2018.

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA. **Tarifas - ANEEL**. Disponível em: <http://www.aneel.gov.br/dados/tarifas>. Acesso em: 2 abr. 2018.

AHMADI, Samad; OSMAN, Ibrahim H. Greedy random adaptive memory programming search for the capacitated clustering problem. **European Journal of Operational Research**, v. 162, n. 1, p. 30–44, 2005.

AL-KHEDHAIRI, Abdulrahman. Simulated annealing metaheuristic for solving p-median problem. **Int. J. Contemp. Math. Sciences**, v. 3, n. 28, p. 1357-1365, 2008.

ALLSTATE ELECTRIC. **EVSE Home Installation & Repair | Allstate Electric**. Disponível em: <https://www.electrician.vegas/residential/evse-installation-repair>. Acesso em: 19 mar. 2018.

ALP, Osman; ERKUT, Erhan; DREZNER, Zvi. An efficient genetic algorithm for the p-median problem. **Annals of Operations research**, v. 122, n. 1-4, p. 21-42, 2003.

ANDREWS, Matthew et al. Modeling and optimization for electric vehicle charging infrastructure. **Ect. Bell-Labs. Com**, n. 2010, p. 1-10, 2012.

ASSOCIAÇÃO BRASILEIRA DO VEÍCULO ELÉTRICO. **Brasil tem frota de somente 25 mil carros elétricos e híbridos**. Disponível em: <http://www.abve.org.br/noticias/brasil-tem-frota-de-so-25-mil-carros-eletricos-e-hibridos>. Acesso em: 8 fev. 2017.

BALDACCI, Roberto et al. A new method for solving capacitated location problems based on a set partitioning approach. **Computers & Operations Research**, v. 29, n. 4, p. 365–386, 2002.

BALINSKI, M L. Integer Programming: Methods, Uses, Computations. **Management Science**, v. 12, n. 3, p. 253–313, 1965.

BAOUCHE, Fouad et al. Efficient Allocation of Electric Vehicles Charging Stations: Optimization Model and Application to a Dense Urban Network. **IEEE Intelligent Transportation Systems Magazine**, v. 6, n. 3, p. 33–43, 2014.

BAOUCHE, Fouad; BILLOT, Romain; EL FAOUZI, Nour-Eddin. Electric Vehicle Charging Stations Allocation Model. **Transport Research Arena**, 2014.

BEASLEY, J. E. A note on solving large p-median problems. **European Journal of Operational Research**, v. 21, n. 2, p. 270–273, 1985.

BEASLEY, J.E. **OR-LIBRARY**. Disponível em: <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>. Acesso em: 25 fev. 2017.

BOCCIA, Maurizio et al. A cut and branch approach for the capacitated p-median problem based on fenchel cutting planes. **Journal of Mathematical Modelling and Algorithms**, v. 7, n. 1, p. 43–58, 2008.

CESELLI, Alberto; RIGHINI, Giovanni. A branch-and-price algorithm for the capacitated p-median problem. **Networks**, v. 45, n. 3, p. 125–142, 2005.

CHAVES, Antonio; DE ASSIS CORREA, Francisco; LORENA, Luiz. Clustering search heuristic for the capacitated p-median problem. **Innovations in Hybrid Intelligent Systems**, p. 136–143, 2007.

CHEN, T. Donna et al. The electric vehicle charging station location problem: a parking-based assignment method for Seattle. In: Transportation Research Board 92nd Annual Meeting, 92., 2013, Washington. 2013. **Anais…** Washington: The National Academies of Sciences, Engineering, and Medicine, 2013. p. 13–1254.

CHURCH, Richard; REVELLE, Charles. The maximal covering location problem. **Papers of the Regional Science Association**, v. 32, n. 1, p. 101–118, 1974.

COBB, Jeff. **Americans Buy Their Four-Millionth Hybrid Car**. Disponível em: <http://www.hybridcars.com/americans-buy-their-four-millionth-hybrid-car/>. Acesso em: 15 mar. 2018.

CONFEDERAÇÃO NACIONAL DO TRANSPORTE. **Anuário CNT do transporte**. 2016. Disponível em: <http://anuariodotransporte.cnt.org.br/2017/>. Acesso em: 14 mar. 2018.

CONNOLLY, David. General purpose simulated annealing. **Journal of the Operational Research Society**, v. 43, n. 5, p. 495–505, 1992.

CORREA, Elon Santos et al. A genetic algorithm for solving a capacitated p-median problem. **Numerical Algorithms**, v. 35, n. 2, p. 373–388, 2004.

DAI, Jing D. et al. **Electric vehicle (EV) charging infrastructure with charging stations optimumally sited**. U.S. Patent n. 9,132,742, 15 set. 2015.

DASKIN, Mark S. A maximum expected covering location model: formulation, properties and heuristic solution. **Transportation science**, v. 17, n. 1, p. 48–70, 1983.

DASKIN, Mark S; MAASS, Kayse Lee. The p-median problem. In: LAPORTE, Gilbert (Ed.). **Location science**. Cham: Springer, 2015. p. 21-45.

DBCITY. **Vitória, Espírito Santo, Brasil - Coordenadas geográficas Vitória**. Disponível em: <http://pt.db-city.com/Brasil--Espírito-Santo--Vitória>. Acesso em: 1 abr. 2018.

DEPARTAMENTO NACIONAL DE TRÂNSITO. **Frota de Veículos do Brasil**. Disponível em: <http://www.denatran.gov.br/index.php/estatistica/237-frota-veiculos>. Acesso em: 8 fev. 2017.

DIAZ, Juan A; FERNANDEZ, Elena. Hybrid scatter search and path relinking for the capacitated p-median problem. **European Journal of Operational Research**, v. 169, n. 2, p. 570–585, 2006.

DRIVE CLEAN. **Drive Clean | Plug-in Electric Vehicle Resource Center**. Disponível em: <https://www.driveclean.ca.gov/pev/Plug-in_Electric_Vehicles/PEV_Types.php>. Acesso em: 15 mar. 2018.

ELECTRIC TRANSPORTATION ENGINEERING CORPORATION. **Electric Vehicle Charging Infrastructure Deployment Guidelines for the State of Tennessee**. 2010. Disponível em: <http://www.10xe.orwww.10xe.org/Content/Files/Tennessee%20EV%20Infrastructure%20%20Guidelines.pdf>. Acesso em: 20 mar. 2016.

EMPRESA DE PESQUISA ENERGÉTICA. **Anuário Estatístico de Energia Elétrica**. 2016. Disponível em: <http://www.epe.gov.br/AnuarioEstatisticodeEnergiaEletrica/Forms/Anurio.aspx>. Acesso em: 8 fev. 2017.

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION. **ECMA-334: ISO/IEC C# Language Specification**, 2006.

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION. **ECMA-262: ECMAScript Language Specification**, 2009.

EVADOPTION. **EV Statistics of the Week: Range, Price and Battery Size of Currently Available (in the US) BEVs - EV Adoption**. Disponível em: <http://evadoption.com/ev-statistics-of-the-week-range-price-and-battery-size-of-currently-available-in-the-us-bevs/>. Acesso em: 20 mar. 2018.

FAIZ, Asif. et al. **Air pollution from motor vehicles : standards and technologies for controlling emissions**. Washington: World Bank Publications, 1996. Disponível em: <https://books.google.com/books?id=Hqsyv_KD0lgC&pg=PA227+>. Acesso em: 15 mar. 2018.

FLESZAR, Krzysztof; HINDI, Khalil S. An effective VNS for the capacitated p-median problem. **European Journal of Operational Research**, v. 191, n. 3, p. 612–622, 2008.

FRADE, Inês et al. Optimal location of charging stations for electric vehicles in a neighborhood in Lisbon, Portugal. **Transportation research record: journal of the transportation research board**, n. 2252, p. 91–98, 2011.

GALVÃO, Roberto Diéguez; RAGGI, Luiz Aurélio. A method for solving to optimality uncapacitated location problems. **Annals of Operations Research**, v. 18, n. 1, p. 225–244, 1989.

GARY, Michael R.; JOHNSON, David S. **Computers and Intractability: A Guide to the Theory of NP-completeness**. New York: WH Freeman and Company, 1979.

GENERAL MOTORS CORP. **2018 Volt: Plug In Hybrid | Electric Hybrid Car**. Disponível em: <http://www.chevrolet.com/electric/volt-plug-in-hybrid>. Acesso em: 16 mar. 2018.

GLOVER, Fred. Future paths for integer programming and links to artificial intelligence. **Computers & operations research**, v. 13, n. 5, p. 533–549, 1986.

GOOGLE INC. **Google Maps APIs | Google Developers**. Disponível em: <https://developers.google.com/maps/?hl=pt-br>. Acesso em: 13 mar. 2018.

HAKIMI, S L. Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph. **Operations Research**, v. 12, n. 3, p. 450–459, 1964.

HANNAN, M.A. et al. Hybrid electric vehicles and their challenges: A review. **Renewable and Sustainable Energy Reviews**, v. 29, p. 135–150, 2014. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84884581568&partnerID=tZOtx3y1>. Acesso em: 9 jul. 2014.

HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search for the p-median. **Location Science**, v. 5, n. 4, p. 207–226, 1997.

INTERNATIONAL BUSINESS MACHINES CORPORATION. **IBM CPLEX Optimizer - United States**. Disponível em: <https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>. Acesso em: 16 fev. 2017.

JAMES, J. Q. et al. Evolutionary artificial neural network based on chemical reaction optimization. In: Evolutionary Computation (CEC) IEEE Congress, 12., 2011, New Orleans. 2011. **Anais…** New Orleans: IEEE, 2011. p. 2083-2090.

JORNAL DO CARRO. **Nissan Leaf será feito no Brasil**. Disponível em: <http://www.estadao.com.br/jornal-do-carro/noticias/carros,nissan-leaf-sera-feito-no-brasil,26521,0.htm>. Acesso em: 9 fev. 2017.

KUBY, Michael J. Programming Models for Facility Dispersion: The p-Dispersion and Maxisum Dispersion Problems. **Geographical Analysis**, v. 19, n. 4, p. 315–329, 1987.

LAM, Albert Y S; LEUNG, Yiu-wing; CHU, Xiaowen. Electric Vehicle Charging Station Placement. **IEEE Smart Grid Comm**, p. 510–515, 2013.

LAM, Albert YS; LEUNG, Yiu-Wing; CHU, Xiaowen. Electric vehicle charging station placement. In: Smart Grid Communications (SmartGridComm) IEEE International Conference, 2013, Vancouver. 2013. **Anais…** Vancouver: IEEE, 2013. p. 510-515.

LAM, Albert Y S; LEUNG, Yiu-Wing; CHU, Xiaowen. Electric vehicle charging station placement: Formulation, complexity, and solutions. **IEEE Transactions on Smart Grid**, v. 5, n. 6, p. 2846–2856, 2014.

LAM, Albert Y S; LI, Victor O K. Chemical Reaction Optimization: A tutorial. **Memetic Computing**, v. 4, n. 1, p. 3–17, 2012.

LAM, Albert Y S; LI, Victor O K; JAMES, J Q. Real-coded chemical reaction optimization. **IEEE Transactions on Evolutionary Computation**, v. 16, n. 3, p. 339–353, 2012.

LAMBERT, Fred. **US has now ~16,000 public electric vehicle charging stations with ~43,000 connectors | Electrek**. Disponível em: <https://electrek.co/2017/06/19/us-electric-vehicle-charging-stations/>. Acesso em: 19 mar. 2018.

LIN, Shen. Computer solutions of the traveling salesman problem. **The Bell system technical journal**, v. 44, n. 10, p. 2245–2269, 1965.

LORENA, Luiz Antonio Nogueira; SENNE, Edson Luiz França. Local search heuristics for capacitated p-median problems. **Networks and Spatial Economics**, v. 3, n. 4, p. 407–419, 2003.

LORENA, Luiz; SENNE, Edson. A column generation approach to capacitated p-median problems. **Computers & Operations Research**, v. 31, n. 6, p. 863–876, 2004.

MANIEZZO, Vittorio; MINGOZZI, Aristide; BALDACCI, Roberto. A bionomic approach to the capacitated p-median problem. **Journal of Heuristics**, v. 4, n. 3, p. 263–280, 1998.

MARANZANA, F E. On the location of supply points to minimize transport costs. **OR**, p. 261–270, 1964.

MCCAULEY, Ryan. **Building Out Electric Vehicle Infrastructure: Where Are the Best Locations for Charging Stations?** Disponível em: <http://www.govtech.com/fs/Building-Out-Electric-Vehicle-Infrastructure-Where-Are-the-Best-Locations-for-Charging-Stations.html>. Acesso em: 19 mar. 2018.

MERZ, Peter; FREISLEBEN, Bernd. A genetic local search approach to the quadratic assignment problem. In: Proceedings of the 7th international conference on genetic algorithms, 7., 1997, Michigan. **Anais…** Michigan: Springer-Verlag. 1997. p. 1-1.

MORROW, Kevin et al. Plug-in hybrid electric vehicle charging infrastructure review. **US Department of Energy-Vehicle Technologies Program**, v. 34, 2008.

MULVEY, John M; BECK, Michael P. Solving capacitated clustering problems. **European Journal of Operational Research**, v. 18, n. 3, p. 339–348, 1984.

NAVIGANT RESEARCH. **Market Data: EV Geographic Forecasts**. Disponível em: <http://www.navigantresearch.com/research/transportation-efficiencies/electric-vehicles>. Acesso em: 14 fev. 2017.

OSMAN, Ibrahim H.; CHRISTOFIDES, Nicos. Capacitated clustering problems by hybrid simulated annealing and tabu search. **International Transactions in Operational Research**, v. 1, n. 3, p. 317–336, 1994.

PLUGINSITES. **Legislation Reference - Reserved Parking for Plug-In Vehicle Charging | PlugInSites**. Disponível em: <http://pluginsites.org/plug-in-vehicle-parking-legislation-reference/>. Acesso em: 30 abr. 2018.

REINELT, Gerhard. **TSPLIB – TSP Data**. Disponível em: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>. Acesso em: 22 mar. 2018.

REPEDE, John F; BERNARDO, John J. Developing and validating a decision support system for locating emergency medical vehicles in Louisville, Kentucky. **European journal of operational research**, v. 75, n. 3, p. 567–581, 1994.

REVISTA PEQUENAS EMPRESAS GRANDES NEGÓCIOS. **Empresas começam a instalar postos de recarga rápida para carros elétricos - PEGN | Tecnologia**. Disponível em: <https://revistapegn.globo.com/Tecnologia/noticia/2017/09/empresas-comecam-instalar-postos-de-recarga-rapida-para-carros-eletricos.html>. Acesso em: 15 mar. 2018.

REVELLE, Charles; HOGAN, Kathleen. The maximum availability location problem. **Transportation Science**, v. 23, n. 3, p. 192–200, 1989.

ROLLAND, Erik et al. An efficient tabu search procedure for the p-Median Problem. **European Journal of Operational Research**, v. 96, n. 2, p. 329–342, jan. 1997. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0377221796001415>. Acesso em: 14 fev. 2017.

ROSING, K. E.; REVELLE, C. S. Heuristic concentration: Two stage solution construction. **European Journal of Operational Research**, v. 97, n. 1, p. 75–86, 1997.

SAXTON, Tom. **Understanding Electric Vehicle Charging « Plug In America**. Disponível em: <https://pluginamerica.org/understanding-electric-vehicle-charging/>. Acesso em: 15 mar. 2018.

SCHEUERER, Stephan; WENDOLSKY, Rolf. A scatter search heuristic for the capacitated clustering problem. **European Journal of Operational Research**, v. 169, n. 2, p. 533–547, 2006.

SCHMIDT, Eric. **2017 Battery Electric Cars Reported Range Comparison**. Disponível em: <https://www.fleetcarma.com/2017-battery-electric-cars-reported-range-comparison/>. Acesso em: 16 mar. 2018.

SHAO-YUN, Ge et al. The Planning of Electric Vehicle Charging Stations in the Urban Area. **2nd International Conference on Electronic & Mechanical Engineering and Information Technology (EMEIT-2012)**, p. 1598–1604, 2012.

SHAO-YUN, G. et al. The planning of electric vehicle charging stations in the urban area. In: **Electric & Mechanical Engineering and Information Technology (EMEIT) Conference**,2., 2012, Shenyang. **Anais…** Red Hook: Curran. 2013. p. 1598-1604.

SILVA, Danilo; MESTRIA, Mário. Chemical Reaction Optimization metaheuristic for locating service stations through the capacitated p- median problem. **Revista Pesquisa Operacional**, 2018 (artigo em revisão).

SILVA, Danilo; MESTRIA, Mário. **Solução do problema da p-mediana para localização de estações de serviço utilizando metaheurística chemical reaction optimization**, 49., 2017, Blumenau. **Anais eletrônicos…** Rio de Janeiro: SOBRAPO. 2017. p. 2113-2124. Disponível em: <http://www.sbpo2017.iltc.br/pdf/168507.pdf>. Acesso em: 3 abr. 2018.

STEFANELLO, Fernando et al. Matheuristics for the capacitated p-median problem. **International Transactions in Operational Research**, v. 22, n. 1, p. 149–167, 2015.

SUTOR, Julie M; HUDGINS, Andrew P. Plug-In Electric Vehicle Handbook for Workplace Charging Hosts (Brochure), Clean Cities, Energy Efficiency & Renewable Energy (EERE). **US Department of Energy**, 2016. Disponível em: <https://www.afdc.energy.gov/uploads/publication/pev_consumer_handbook.pdf>. Acesso em: 15 mar. 2018.

TOREGAS, Constantine et al. The Location of Emergency Service Facilities. **Operations Research**, v. 19, n. 6, p. 1363–1373, 1971. Disponível em: <http://dx.doi.org/10.1287/opre.19.6.1363>.

TRIGUI, Rochdi et al. Systemic modelling of hybrid vehicles in order to predict dynamic performance and energy consumption building the VEHLIB library of models. **RTS-Recherche Transports Securite**, n. 83, p. 129–150, 2004.

TURCHETTA, Diane. **Public Roads - The Car of The Future, Today, November/December 2012 - FHWA-HRT-13-001**. Disponível em: <https://www.fhwa.dot.gov/publications/publicroads/12novdec/01.cfm>. Acesso em: 16 mar. 2018.

US ENERGY DEPARTAMENT. **PHEV Battery Testing Results - 2013 Toyota Prius - VIN 6237**. 2013. Disponível em: <https://www.energy.gov/sites/prod/files/2015/03/f20/batteryPrius6237.pdf>. Acesso em: 15 mar. 2018.

WANG, Ying-Wei. An optimal location choice model for recreation-oriented scooter recharge stations. **Transportation Research Part D: Transport and Environment**, v. 12, n. 3, p. 231–237, 2007.

WANG, Ying-Wei. Locating battery exchange stations to serve tourism transport: A note. **Transportation Research Part D: Transport and Environment**, v. 13, n. 3, p. 193–197, 2008.

WANG, Ying-Wei; LIN, Chuah-Chih. Locating road-vehicle refueling stations. **Transportation Research Part E: Logistics and Transportation Review**, v. 45, n. 5, p. 821–829, 2009.

WANG, Ying-Wei; WANG, Chuan-Ren. Locating passenger vehicle refueling stations. **Transportation Research Part E: Logistics and Transportation Review**, v. 46, n. 5, p. 791–801, 2010.

WHITAKER, R A. A fast algorithm for the greedy interchange for large-scale clustering and median location problems. **INFOR: Information Systems and Operational Research**, v. 21, n. 2, p. 95–108, 1983.

WU, Chunyang et al. A method for electric vehicle charging infrastructure planning. **Dianli Xitong Zidonghua/Automation of Electric Power Systems**, v. 34, n. 24, 2010.

XI, Xiaomin; SIOSHANSI, Ramteen; MARANO, Vincenzo. Simulation--optimization model for location of a public electric vehicle charging infrastructure. **Transportation Research Part D: Transport and Environment**, v. 22, p. 60–69, 2013.

XU, Jin et al The 2010 World Congress in Computer Science, Computer Engineering, and Applied Computing (Worldcomp 2010), 5., 2010, Las Vegas. **Anais…** Las Vegas:Worldcomp. 2010. p. 125-131.

APÊNDICE A – Implementação dos modelos de localização no CPLEX

O modelo matemático de otimização, descrito na etapa 3.2, foi implementado em linguagem *Optimization Programming Language* (OPL) (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) para execução pelo otimizador CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017). Além de executar modelos, o CPLEX também conta com um ambiente integrado de desenvolvimento, denominado *IBM ILOG CPLEX Optimization Studio*. A Figura 29 ilustra a aparência geral do ambiente.

O CPLEX trabalha com o conceito de projetos. Cada projeto compõe-se de um ou mais componentes contendo modelos em OPL, dados e configurações. Os mesmos estão descritos a seguir:

1 MODELOS DE OTIMIZAÇÃO

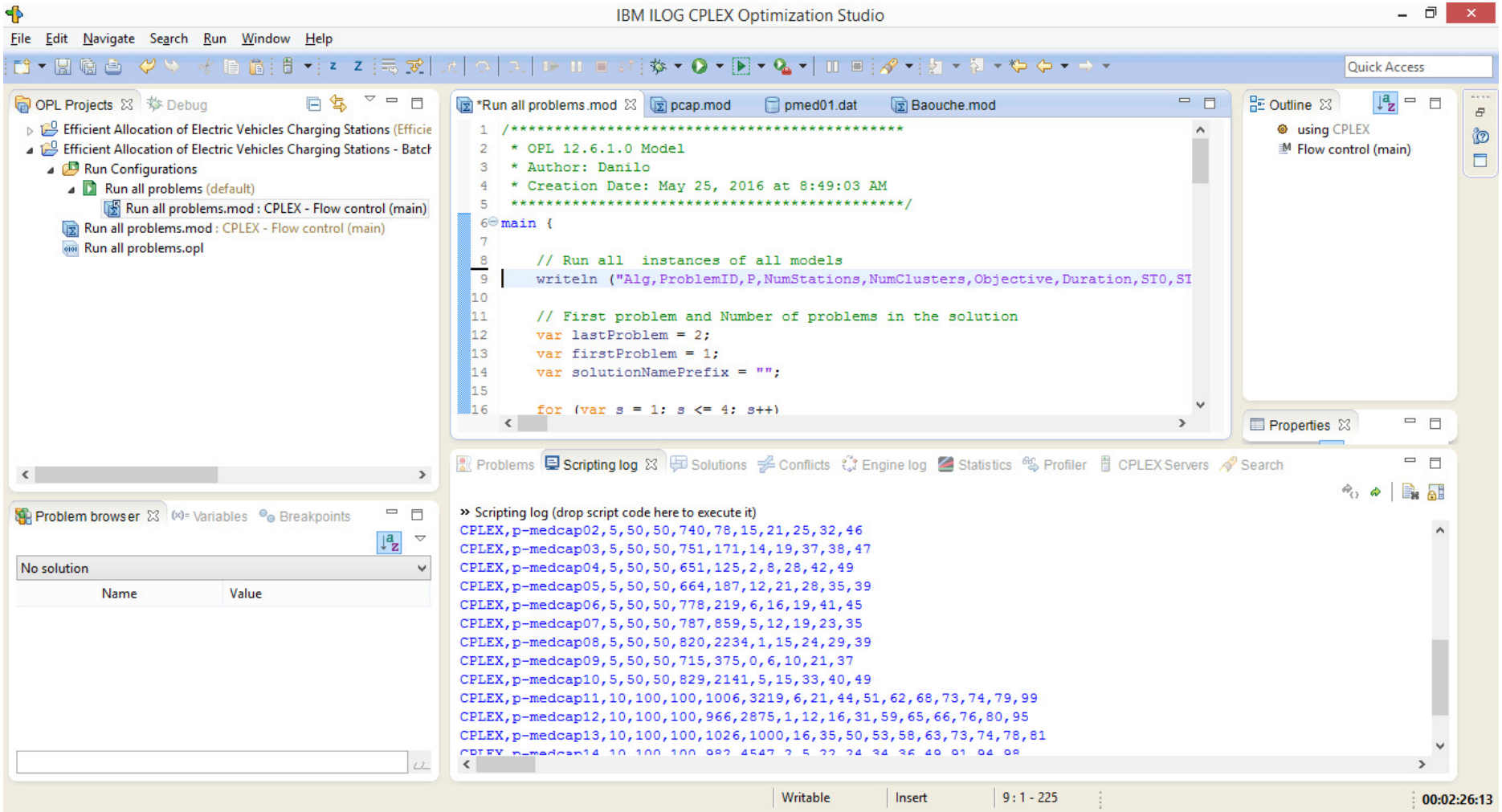
Cada projeto contém um ou mais arquivos, com extensão *.mod,* com modelos de otimização, escritos em linguagem OPL. Arquivos do tipo *.mod* podem conter também *scripts* para controle de fluxo, que permitem efetuar um pré-processamento ou pós-processamento dos dados a serem processados pelo modelo OPL. Tais *scripts* não são codificados em OPL, mas numa implementação proprietária da linguagem de programação *JavaScript* (EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION, 2009), denominada IBM ILOG *Script*, voltada a este tipo de processamento. A Figura 12 ilustra a implementação em linguagem OPL do modelo do modelo de Baouche e colaboradores (2014).

No presente trabalho, além de modelos em OPL para resolução do modelo de Baouche e colaboradores (2014) e dos modelos da p-mediana capacitada e não-capacitada, foram também criados scripts de pré-processamento para execução em lote de todos os modelos matemáticos utilizados na dissertação, tendo como entrada um ou mais problemas das bases de dados descritas na etapa 3.3. A Figura 30 contém o código de pré-processamento parcial em *Javascript*, responsável pela execução em lote dos problemas.

Por fim, para que saída de dados pudesse ser efetuada no formato descrito na etapa 3.4, também foram criados scripts para pós-processamento dos resultados da execução dos modelos.

A Figura 31 mostra o código parcial do script de pós-processamento utilizado para tal finalidade.

Figura 29 – *IBM ILOG CPLEX Optimization Studio*



Fonte: Elaborado pelo autor (2017)

Figura 30 – Código para pré-processamento de modelos em *Javascript*

```
/*********************************************
 * OPL 12.6.1.0 Model
 * Author: Danilo Silva
 * Creation Date: May 25, 2016 at 8:49:03 AM
 *********************************************/
main
{
    // Run instances of models in batch mode
    writeln ("Alg,ProblemID,P,NumStations,NumClusters,Objective,…
    var lastProblem = 1;
    var firstProblem = 1;
    var solutionNamePrefix = "";
    for (var s = 1; s <= 5; s++)
    {
        // pmed
        if (s==1)
        {
            firstProblem = 1;
            lastProblem = 40;
            solutionNamePrefix = "pmed"
        }
        // pmed Baouche
        else if (s==2)
        {
            firstProblem = 1;
            lastProblem = 20;
            solutionNamePrefix = "baouchepmed"
        }
...
        // vix
        else if (s==5)
        {
            firstProblem = 1;
            lastProblem = 5;
            solutionNamePrefix = "vix"
        }
        else
            writeln ("Invalid solution!")

        for (var i = firstProblem; i <= lastProblem; i++)
        {

            var proj = new IloOplProject("C:\\... Vehicles Charging Stations");
            var solutionNumber = "" + parseInt(i);
            solutionNumber = ("0").substring (0,2-solutionNumber.length)+
        solutionNumber;
            var solutionName = solutionNamePrefix + solutionNumber;
            write ("CPLEX,", solutionName, ",")
            var rc = proj.makeRunConfiguration(solutionName);
            rc.oplModel.generate();
            if (rc.cplex.solve())
                rc.oplModel.postProcess();
            else
                writeln("No solution");

            rc.end();
            proj.end();
        }
    }
}
```

Fonte: Elaborado pelo autor (2017)

Figura 31 – Código para pós-processamento de modelos em *Javascript*

```
// Post processing
execute
{   // Objective, Number of instances and Number of Stations
    var ct = 0;
    for (var i in stations)
        {
            if ( x[i] == 1 ) ct++
        };
    write (ct, ",");
    write (numStations, ",");
    write (numClusters, ",");
    write (cplex.getObjValue(), ",");

    // Duration
    write (cplex.getSolvedTime()*1000.0,",");

    // Selected stations
    var ct1 = 0;
    for (var i in stations)
    {
        if ( x[i] == 1 )
        {
            write (i-1);
            ct1++;
            write (",");
        }
    };

    // Max distance between consumer and station
    var maxDist = 0;
    for (var i in clusters)
    {
        for (var j in clusters)
        {
             if (y[i][j] == 1)
               if (dist[i][j] > maxDist)
                   maxDist = dist[i][j]

        };
    }
    write(maxDist, ",")

    // Min distance between stations
    var minDist = M;
    for (var i in clusters)
    {
        for (var j in clusters)
        {
            if ((x[i]== 1) && ( x[j] == 1 ) && (i != j) )
                if (dist [i][j] < minDist)
                    minDist = dist [i][j]
        };
    }
    write(minDist, ",");
    writeln ();
}
```

Fonte: Elaborado pelo autor (2017)

2 ARQUIVOS DE DADOS

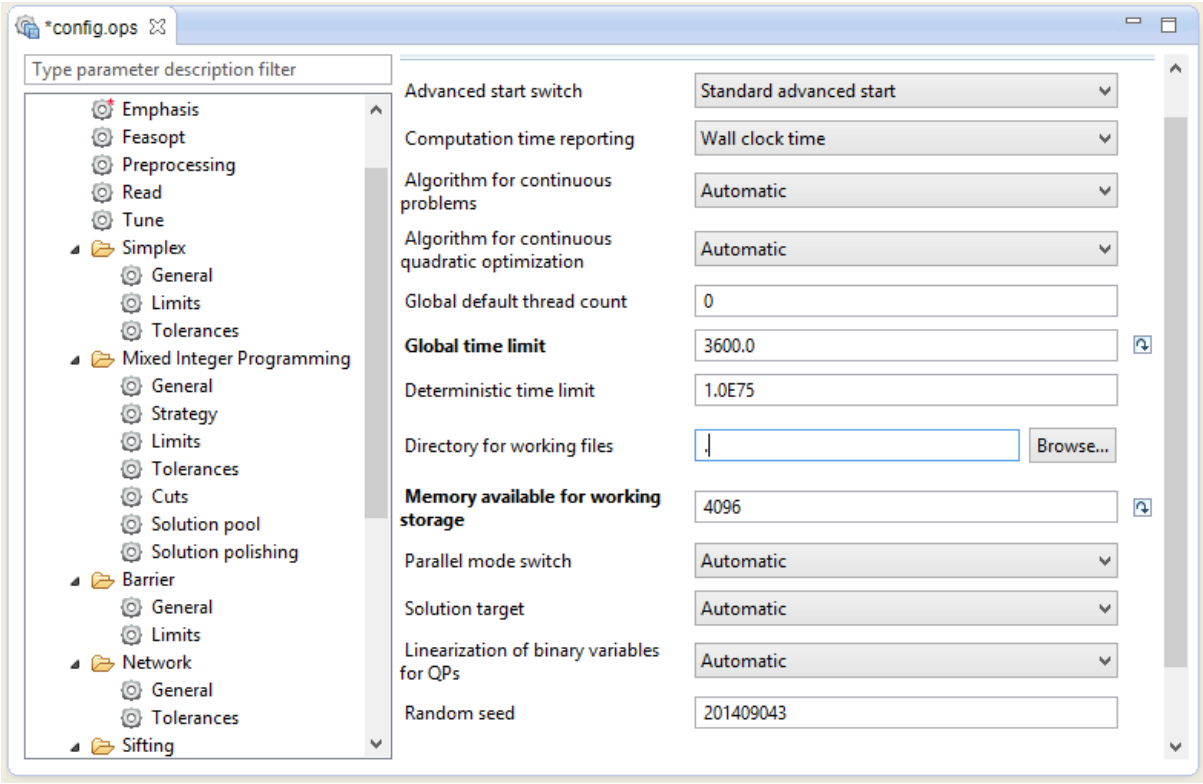
Projetos OPL podem conter, opcionalmente, arquivos de dados, com extensão *.dat*, que contêm dados a serem processados pelos modelos de otimização, escritos em linguagem OPL, de modo a permitir que tais modelos não precisem conter dados estáticos pertencentes a problemas. Tal funcionalidade permite a independência entre modelo de otimização e os dados das instâncias. A Figura 10 mostra um dos arquivos de dados utilizados nesta dissertação, correspondente à instancia *sjc1*, criada por Lorena e Senne (2004).

No presente trabalho, um total de 65 arquivos de dados foram criados, contendo instâncias de p-mediana da biblioteca *OR-Library* (BEASLEY, 1990) e das instâncias criadas na Etapa 3.7. Um programa de importação e transformação de dados foi criado para converter os dados dos problemas de seus formatos originais para um formato comum, aceito pelo otimizador CPLEX, conforme descrito na Etapa 3.4.

3 ARQUIVOS DE CONFIGURAÇÃO

O terceiro componente, também opcional, de um projeto é o arquivo de configuração. Utiliza a extensão *.ops,* e contém configurações de execução diferentes daquelas que são utilizadas, por padrão, pelo CPLEX durante a resolução de problemas. Os parâmetros de configuração que podem ser modificados incluem os tipos de algoritmos usados para resolver os modelos, memória máxima e tempo máximo de execução. A Figura 32 mostra um arquivo de configuração típico, onde o tempo máximo global de execução (*Global time limit*) foi alterado para 1 hora (3600s) e a quantidade de memória total utilizada pelo otimizador (*Memory available for working storage)* foi limitada em 4GB (4096MB).

Figura 32 – Arquivo de configuração OPL típico



Fonte: Elaborado pelo autor (2017)

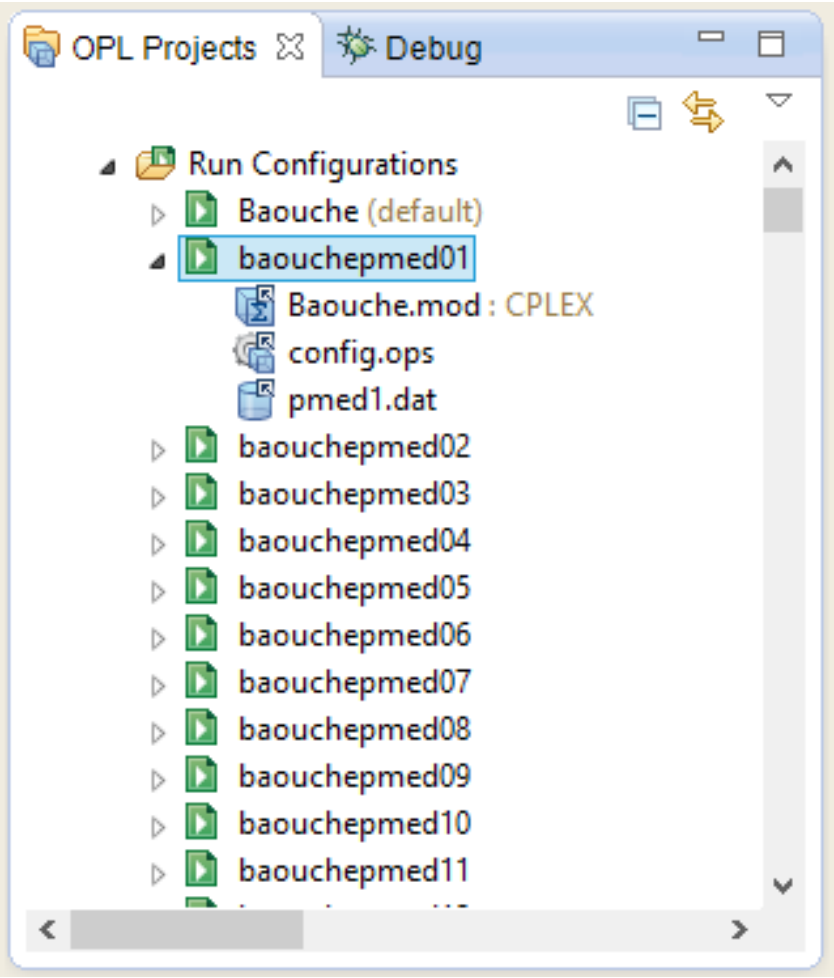
4 CONFIGURAÇÕES DE EXECUÇÃO

Configurações de execução são elementos de um projeto OPL que agrupam um arquivo *.opl*, contendo um modelo matemático, com um arquivo *.dat*, contendo dados a serem processados e, opcionalmente, um arquivo *.ops*, contendo configurações de execução diferentes do padrão do CPLEX. Isto permite que um mesmo modelo matemático possa ser resolvido para diferentes problemas (dados de entrada). A Figura 33 ilustra uma configuração de execução típica.

Na presente dissertação, 195 configurações de execução foram criadas, que correspondem ao produto dos 3 modelos codificados em OPL (p-mediana não-capacitada, p-mediana capacitada e modelo proposto) pelos 65 problemas resolvidos de forma exata pelo CPLEX. Isto possibilitou que qualquer modelo pudesse ser resolvido com qualquer um dos problemas considerados.

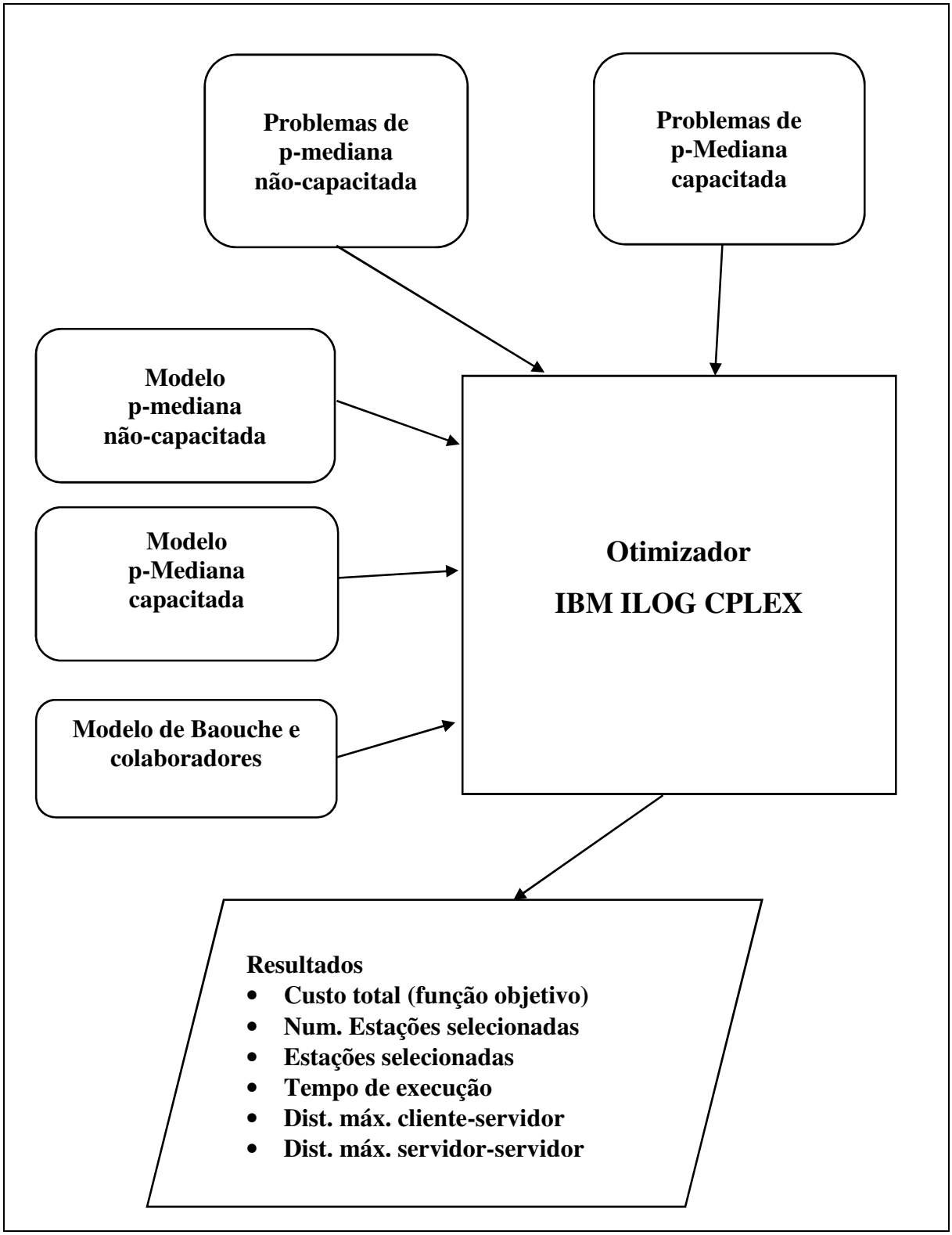
A Figura 34 mostra um fluxograma com as entradas e saídas dos projetos OPL.

Figura 33 – Configuração de execução de um projeto OPL



Fonte: Elaborado pelo autor (2017)

Figura 34 – Fluxograma dos projetos OPL



Fonte: Elaborado pelo autor (2017)

5 VERIFICAÇÃO DA EXATIDÃO DA CODIFICAÇÃO DOS MODELOS OPL

As bases de dados da *OR-Library*, após importadas e convertidas para formatos aceitos pelo CPLEX, foram utilizadas para verificar a exatidão da codificação, em linguagem OPL, dos três modelos utilizados no presente trabalho. Para que os dois modelos de p-mediana houvessem sido codificados corretamente, os valores ótimos para a função objetivo, obtidos na resolução de tais modelos, para os diferentes problemas contidos nas bases de dados, deveriam ser idênticos aos reportados na literatura, o que de fato ocorreu. Os tempos de execução também foram registrados, para posterior comparação com os obtidos através do programa, em linguagem C#, que implementa a metaheurística CRO.

Embora a codificação dos modelos em linguagem OPL tenha sido relativamente simples, consistindo na transcrição direta da função objetivo e restrições de cada modelo, reduzindo assim a possiblidade de ocorrência de erros de programação, alguns testes básicos foram conduzidos, com o objetivo de avaliar o comportamento do modelo proposto por Baouche e colaboradores (2014), uma vez que valores de referência não estavam disponíveis na literatura, como nos modelos de p-mediana.

Para as bases de dados da p-mediana *não-capacitada* da *OR-Library*, uma distância de separação mínima entre estações de 20Km foi estabelecida, a fim de verificar-se o comportamento da função objetivo e do número de estações abertas, em função da constante de p-dispersão, *r*. A Tabela 13 exibe os resultados do processamento, pelo CPLEX, dos problemas de p-mediana não-capacitada da *OR-Library*, utilizando o modelo de p-mediana não capacitada e o modelo de Baouche e colaboradores (2014). A tabela mostra a número do problema (*Prob*) e o número de estações candidatas / clientes (*n*). Nas colunas subsequentes são exibidos, para cada modelo, o valor ótimo da função objetivo, o número de medianas (*p*), a distância máxima entre cliente e estação e a distância mínima entre estações.

No modelo da p-mediana, a distância máxima de um agrupamento de clientes à estação selecionada para atendê-los, excedeu a 50 Km (metade da autonomia média de um EV típico adicionado de uma margem de segurança de 40%) em 47% dos problemas resolvidos. No modelo de Baouche e colaboradores (2014), a distância máxima de um agrupamento de clientes a estação que foi selecionada para atendê-los, excedeu a 50 Km em somente 23% dos problemas resolvidos. No modelo da p-mediana, a distância de separação entre estações de carregamento foi inferior a 20 Km em 76% dos problemas resolvidos, enquanto que no modelo de Baouche e

colaboradores (2014) em 100% dos problemas, a distância esteve sempre acima de 20Km, conforme especificado na restrição de p-dispersão do modelo (7). No modelo de Baouche e colaboradores (2014), o número de estações alocadas foi maior que no modelo p-mediana em 30 dos 40 problemas resolvidos, o que deve ser atribuído ao fato de que o custo de implantação das estações, presente no modelo de Baouche e colaboradores (2014), ter sido feito igual a zero, por não fazer parte do modelo da p-mediana não-capacitada.

Para a avaliação do comportamento do modelo proposto por Baouche e colaboradores (2014), com as bases de dados da p-mediana capacitada da *OR-Library*, a mesma distância de separação mínima entre estações de 20Km foi mantida. Os resultados obtidos para os modelos da p-mediana capacitada e modelo de Baouche e colaboradores (2014), utilizando os problemas de p-mediana capacitada da base de dados *OR-Library*, estão listados na Tabela 14. A tabela mostra a número do problema (*Prob*) e o número de estações candidatas / clientes (*n*). Nas colunas subsequentes são exibidos, para cada modelo, o valor ótimo da função objetivo, o número de medianas (*p*), a distância máxima entre cliente e estação e a distância mínima entre estações.

Em nenhum dos modelos e problemas resolvidos de p-mediana capacitada, a distância máxima de um agrupamento de clientes à estação que foi selecionada para atendê-los, excedeu a 50 Km. No modelo p-mediana capacitada, a distância de separação entre estações de carregamento foi inferior a 20 Km em 30% dos problemas resolvidos. No modelo de Baouche e colaboradores (2014), o número de estações alocadas foi sempre maior que no modelo p-mediana, o que pode ser função de ter-se atribuído zero ao custo de implantação das estações, de modo a manter compatibilidade com o modelo da p-mediana capacitada, uma vez que o mesmo não o possui em sua definição.

Uma conclusão que se chegou, a partir dos testes realizados, é a de que, para se limitar de forma efetiva o número de estações de carregamento abertas a uma quantidade razoável, deve-se atribuir uma capacidade limite de atendimento de EVs às estações candidatas, bem como estabelecer os custos de implantação das mesmas, conforme modelo da Etapa 3.2. Assim, restringir a capacidade de fornecimento de energia das estações poderá aumentar o número de estações selecionadas. De modo contrário, quanto maior o custo de instalação das estações, tanto menor será o número de estações selecionadas.

Adicionalmente, deve existir sempre um afastamento mínimo, *r*, entre estações de carregamento, a fim de reduzir o número de estações selecionadas, já que no modelo de Baouche e colaboradores (2014) o número de estações (medianas) não é fixo.

Tabela 13 – Comparação entre p-mediana *não-capacitada* e modelo de Baouche e colab.

		p-mediana não-capacitada				Baouche e colab.(2014)			
Prob	***n***	**Valor ótimo da função Objetivo**	***p***	**Dist. Max. Cliente-Est.**	**Dist. Mín. entre Est.**	**Valor ótimo da função Objetivo**	***p***	**Dist. Max. Cliente-Est.**	**Dist. Mín. entre Est.**
1	100	5819	5	133	81	242	75	19	20
2	100	4093	10	132	50	216	72	18	20
3	100	4250	10	186	38	277	73	19	20
4	100	3034	20	92	36	235	78	19	20
5	100	1355	33	53	23	256	66	17	20
6	200	7824	5	101	39	719	109	19	20
7	200	5631	10	77	14	841	110	19	20
8	200	4445	20	102	24	693	119	19	20
9	200	2734	40	58	14	710	112	19	20
10	200	1255	67	23	10	910	93	19	20
11	300	7696	5	69	13	1505	128	18	20
12	300	6634	10	86	21	1559	127	19	20
13	300	4374	30	56	8	1525	133	19	20
14	300	2968	60	41	10	1450	129	19	20
15	300	1729	100	21	10	1456	130	19	20
16	400	8162	5	52	18	2363	126	19	20
17	400	6999	10	50	12	2457	132	19	20
18	400	4809	40	54	12	2205	151	19	20
19	400	2845	80	26	7	2425	131	19	20
20	400	1789	133	17	5	2381	135	19	20
21	500	9138	5	44	16	3299	135	19	20
22	500	11296	10	67	4	16035	24	55	21
23	500	4619	50	37	7	4735	96	32	20
24	500	2961	100	23	6	4136	103	19	20
25	500	1828	167	14	3	4747	93	32	20
26	600	9917	5	48	12	10726	38	43	20
27	600	8307	10	46	13	21622	14	66	46
28	600	4498	60	31	4	22869	15	61	33
29	600	3033	120	19	4	10150	36	39	21
30	600	1989	200	12	4	23933	16	62	43
31	700	10086	5	38	10	29233	4	60	51
32	700	9297	10	87	9	27069	7	61	47
33	700	4700	70	30	5	35617	6	74	56
34	700	3013	140	16	4	6370	104	19	20
35	800	10400	5	37	10	8898	64	28	20
36	800	9934	10	45	11	7612	98	26	20
37	800	5057	80	29	4	9718	59	28	20
38	900	11060	5	44	11	10700	52	28	20
39	900	9423	10	83	8	12308	32	32	20
40	900	5128	90	25	4	9081	89	19	20

Fonte: Elaborado pelo autor (2017)

Tabela 14 – Comparação entre p-mediana capacitada e modelo de Baouche e colab.

		p-mediana capacitada				Baouche e colab.(2014)			
Prob	***n*.**	**Valor ótimo função objetivo**	***p***	**Dist. Max. Cliente-Est.**	**Dist. Mín. entre Est.**	**Valor Função Obj.**	***p***	**Dist. Max. Cliente Est.**	**Dist. Mín. entre Est.**
1	50	713	5	50	35	312	16	16	20
2	50	740	5	43	21	279	17	16	21
3	50	751	5	32	25	313	15	17	20
4	50	651	5	42	34	290	14	17	20
5	50	664	5	39	36	330	15	18	20
6	50	778	5	39	31	305	18	16	20
7	50	787	5	48	32	356	15	19	20
8	50	820	5	35	25	242	16	19	20
9	50	715	5	39	27	336	15	16	20
10	50	829	5	40	33	314	17	16	20
11	100	1006	10	23	21	684	19	18	20
12	100	966	10	30	14	657	20	16	20
13	100	1026	10	26	19	708	19	19	20
14	100	982	10	32	18	729	20	18	20
15	100	1091	10	32	22	686	21	20	20
16	100	954	10	29	16	740	19	19	20
17	100	1034	10	31	18	694	21	22	20
18	100	1043	10	26	18	668	21	18	20
19	100	1031	10	26	20	713	20	20	20
20	100	1005	10	29	21	686	18	19	20

Fonte: Elaborado pelo autor (2017)

APÊNDICE B – Metaheurística CRO para a p-mediana não-capacitada

Os itens a seguir estão descritos, resumidamente, os principais componentes do CRO, bem como os detalhes particulares da presente implementação para os problemas da p-mediana não-capacitada, cujo modelo está descrito na etapa 3.3.1. Os resultados computacionais para a p-mediana clássica são apresentados na seção 4. Maiores informações sobre a metaheurística CRO podem ser obtidas em Lam e Li (2012).

1 MOLÉCULA

O CRO é um algoritmo multiagente, cujos agentes manipulados são moléculas. A molécula é a unidade básica do CRO, e contém vários atributos requeridos para as operações do CRO:

- Estrutura molecular (ω): armazena uma solução do problema, ou seja, um valor para a função objetivo, além das variáveis de decisão x e y do problema da p-mediana. As variáveis x e y foram implementadas como listas encadeadas de inteiros;

- Energia potencial (*PE*): é definida como o valor da função objetivo da solução correspondente, representada por ω. Se f denota a função objetivo, então tem-se que: $\boldsymbol{PE\omega = f(\omega)}$

- Energia cinética (*KE*): é um número não negativo que quantifica a tolerância do sistema em aceitar uma solução pior do que a existente;

- Número de colisões (*NumHit*): número total de colisões que uma molécula já sofreu;

- Estrutura mínima (*MinStruct*): é a solução (ω) com energia potencial (*PE*) mínima, que uma molécula atingiu, até agora. Preserva a estrutura com o menor *PE* no histórico de reações;

- Energia potencial mínima (*MinPE*): é o *PE* correspondente ao *MinStruct*;

- Número mínimo de colisões (*MinHit*): é o número da colisão em que a molécula atingiu o menor valor de *PE* (*MinPE*).

2 REAÇÕES ELEMENTARES

Existem quatro tipos de reações elementares que podem ocorrer a cada iteração do CRO. Elas são empregadas para manipular soluções (explorar o espaço de solução) e redistribuir energia

entre as moléculas e o *buffer* de energia. Operadores são usados para modificar as soluções ou gerar novas soluções a partir das soluções correntes. No entanto, o CRO sempre garante a conservação da energia quando novas soluções são geradas através dos operadores.

Para as colisões ineficazes com a parede e as intermoleculares, foi utilizado um operador baseado no algoritmo ***Fast Interchange***, proposto por Whitaker (1983). Nas sínteses, um operador do tipo *crossover* comumente empregado em Algoritmos Genéticos (GA) foi utilizado. Finalmente, nas decomposições foi empregado o operador ***Half-total change***. Como o nome sugere, uma nova solução é gerada partir de uma solução existente, mantendo metade dos valores (estações selecionadas) da solução corrente e atribuindo novos valores à metade restante. A seguir estão descritas as reações elementares previstas no CRO, bem como detalhes de sua implementação no presente trabalho:

2.1 Colisão ineficaz com a parede

Representa a situação na qual uma molécula colide com uma parede do recipiente e é, então, rebatida, permanecendo uma única molécula. Nessa colisão, a solução existente ω é perturbada transformando-se em ω', isto é:

$$\omega \rightarrow \omega'$$

Isso é feito gerando-se um ω' que esteja na vizinhança de ω, através do operador *Fast Interchange* (FI). Seja *FI(.)* um operador de busca em vizinhança, do tipo *Fast Interchange*. Assim, tem-se que $\omega' = FI(\omega)$ e $PE\omega' = f(\omega')$. Nesse tipo de reação, tipicamente, ocorrerá uma perda de energia potencial, ou seja, $PE\omega'$ será menor que $PE\omega$, indicando que uma solução melhor foi obtida. Caso isto não ocorra e $PE\omega'$ seja maior que $PE\omega$, ainda assim a solução pior poderá ser aceita desde que $PE\omega + KE\omega \geq PE\omega'$. No entanto, toda vez que uma reação elementar ocorre, uma certa quantidade de energia cinética (*KE*) é transferida para o *buffer* de energia, diminuindo as chances de que soluções piores sejam aceitas à medida que as iterações ocorrem. A quantidade de energia cinética da molécula obtida a partir da reação ineficaz é indiretamente controlada pelo parâmetro *KElossRate,* que é um valor entre 0 e 1, inclusive, e determina a quantidade mínima de energia cinética que será aproveitada da solução original (ω). Por exemplo, se à *KElossRate* for atribuído o valor 0,8, um mínimo de 80% da energia cinética que a molécula possuía antes da colisão será transferida para a molécula resultante, após a colisão. A diferença de energia cinética será, então, transferida para o *buffer* central de energia. O pseudocódigo para a Colisão ineficaz com a parede está mostrado na Figura 35.

Figura 35 – Pseudocódigo da Colisão ineficaz com a parede

```
Algorithm OnwallIneffectiveCollision
Input: molecule Mω
ω' ← N(ω)
PEω' ← f (ω')
NumHitω ←NumHitω + 1
if PEω + KEω ≥ PEω' then
    Generate a in [KELossRate, 1]
    KEω' ← (PEω − PEω' + KEω) × a
    buffer ← buffer +(PEω − PEω' + KEω)×(1 − a)
    ω ← ω'
    PEω ← PEω'
    KEω ← KEω'
    if PEω < MinPEω then
        MinStructω ← ω
        MinPEω ← PEω
        MinHitω ← NumHitω
end if
end if
```

Fonte: Lam e Li (2012)

2.2 Decomposição

Ocorre quando uma molécula (ω) colide com uma parede e, em seguida, quebra-se em duas partes, produzindo ω'_1 *e* ω'_2, ou seja:

$$\omega \rightarrow \omega'_1 + \omega'_2$$

O objetivo da decomposição é permitir que o sistema explore outras regiões do espaço de solução, após ter efetuado considerável busca local através de colisões ineficazes. No presente trabalho, utilizou-se o algoritmo *Half-total change* para gerar novas soluções. Uma vez que são criadas mais soluções, a soma total de *PE* e *KE* da molécula original pode não ser suficiente. Em outras palavras, pode ocorrer que $PE\omega + KE\omega < PE\omega'_1 + PE\omega'_2$. Como a conservação de energia não é satisfeita nestas condições, esta decomposição deve ser abortada. Para aumentar a chance de ter-se uma decomposição concluída, uma pequena porção de energia do *buffer* é retirada para apoiar a mudança. O pseudocódigo da decomposição está mostrado na Figura 36.

Figura 36 – Pseudocódigo da Decomposição

```
Algorithm 3 Decomposition
Input: molecule Mω
Create Mω'1 and Mω'2
Obtain ω'1 and ω'2 from ω
PEω'1 ← f (ω'1) and PEω'2 ← f (ω'2)
if PEω + KEω ≥ PEω'1 + PEω'2 then
    Edec ← PEω + KEω −(PEω'1 + PEω'2)
    goto label lbl1
else
    Generate δ1, δ2 IN[0, 1]
    Edec ← PEω + KEω + δ1δ2 × buffer − (PEω'1 + PEω'2)
    if Edec ≥ 0 then
        buffer ← buffer × (1 − δ1δ2)
Lbl1:       Generate δ3 IN[0, 1]
        KEω'1 ← Edec × δ3 and KEω'2 ← Edec ×( 1 − δ3)
        MinStructω'1 ← ω'1 and MinStructω'2 ← ω'2
        MinPEω'1 ← PEω'1 and MinPEω'2 ← PEω'2
        Destroy Mω
    else
        NumHitω ← NumHitω + 1
        Destroy Mω'1 and Mω'2
    end if
end if
```

Fonte: Lam e Li (2012)

2.3 Colisão ineficaz intermolecular

Ocorre quando múltiplas moléculas colidem umas com as outras e, em seguida, se afastam. O número de moléculas permanece inalterado, ou seja:

$$\omega_1 + \omega_2 \rightarrow \omega'_{1'} + \omega'_2$$

Esta reação elementar é muito semelhante à colisão ineficaz unimolecular. ω'_1 *e* ω'_2 são obtidos através de $\omega'_1 = FI(\omega_1)$ *e* $\omega'_2 = FI(\omega_2)$. O gerenciamento de energia é semelhante ao da colisão ineficaz com a parede, mas não envolve o *buffer*. Uma vez que mais moléculas estão envolvidas, a soma total de energia do subsistema molecular é maior do que a da colisão ineficaz com a parede. Assim, a probabilidade de as moléculas explorarem seu ambiente imediato é maior, já que as mesmas têm maior flexibilidade para serem transformadas em estruturas moleculares

mais diversas. No presente trabalho utilizou-se o mesmo operador *Fast Interchange*, aplicando-se o operador a cada molécula para obter-se uma nova. O pseudocódigo da Colisão ineficaz intermolecular está mostrado na Figura 37.

Figura 37 – Pseudocódigo da Colisão ineficaz intermolecular

```
Algorithm IntermolecularIneffectiveCollision
Input: molecules Mω1 and Mω2
ω'1 ← N(ω1) and ω'2 ← N(ω2)
PEω'1 ← f(ω'1) and PEω'2 ← f(ω'2)
NumHitω1 ← NumHitω1 + 1 and NumHitω2 ← NumHitω2 + 1
Einter ← (PEω1 + PEω2 + KEω1 + KEω2) − (PEω'1 + PEω'2)
if Einter ≥ 0 then
    Generate delta4 in [0, 1]
    KEω'1 ← Einter×delta4 and KEω'2 ← Einter×(1−delta4)
    ω1 ← N(ω'1) and ω2 ← N(ω'2)
    PEω1 ← PEω'1 and PEω2 ← PEω'2
    KEω1 ← KEω'1 and KEω2 ← KEω'2
    if PEω1 < MinPEω1 then
        MinStructω1 ← ω1
        MinPEω1 ← PEω1
        MinHitω1 ← NumHitω1
    end if
    if PEω2 < MinPEω2 then
        MinStructω2 ← ω2
        MinPEω2 ← PEω2
        MinHitω2 ← NumHitω2
    end if
end if
```

Fonte: Lam e Li (2012)

2.4 Síntese

A síntese é o oposto da decomposição. Uma síntese acontece quando duas moléculas colidem uma contra a outra e se fundem, isto é:

$$\omega_1 + \omega_2 \rightarrow \omega'$$

Nessa reação, permite-se uma mudança muito maior para ω', em relação a ω_1 e ω_2, bem como um aumento considerável da energia cinética da molécula resultante. Assim a mesma possui uma maior "capacidade" de explorar sua região de soluções, devido a sua energia cinética

superior. A ideia por trás da síntese é a diversificação de soluções. No presente trabalho, um operador *crossover* comumente empregado em Algoritmos Genéticos (GA) foi utilizado. O pseudocódigo da síntese está mostrado na Figura 38.

Figura 38 – Pseudocódigo da Síntese

```
Algorithm 5 Synthesis
Input: molecules Mω1 and Mω2
Create Mω'
Obtain ω' from ω1 and ω2
PEω' ← f (ω')
if PEω1 + PEω2 + KEω1 + KEω2 ≥ PEω' then
    KEω' ← (PEω1 + PEω2 + KEω1 + KEω2)− PEω'
    MinStructω' ← ω'
    MinPEω'1 ← PEω'
    Destroy Mω1 and Mω2
else
    NumHitω1 ←NumHitω1 + 1 and NumHitω2 ← NumHitω2 + 1
    Destroy Mω'
end if
```

Fonte: Lam e Li (2012)

3 CONSERVAÇÃO DE ENERGIA

Um dos pressupostos fundamentais do CRO é a conservação da energia, o que significa que a energia não pode ser criada e nem destruída. O sistema, em sua totalidade, compreende a energia de todas as moléculas definidas somada a energia contida no recipiente, que está associada ao *buffer*. A quantidade total de energia de todo o sistema é determinada pelos valores da função objetivo, isto é, a *PE* da população inicial de moléculas, cujo tamanho é *PopSize*, o *KE* inicial (*InitialKe*) atribuído às moléculas e valor inicial de energia do *buffer*.

4 INICIALIZAÇÃO

Inicialmente, foram criadas variáveis escalares correspondentes aos parâmetros operacionais do CRO: ***PopSize, KELossRate, MoleColl, buffer, InitialKE, α*** **e** ***β***. A seguir, definiu-se a classe "Molécula" com seus atributos ***ω, PE, KE, NumHit, MinStruct, MinPE*** **e** ***MinHit*****,** o seu construtor de classe, além de quatro métodos correspondentes às reações elementares, conforme ilustrado na Figura 39. O construtor deve gerar uma solução inicial viável a partir dos dados do problema, ou seja, que não viole as restrições do modelo da p-mediana não-capacitada,

ilustrado no item 3.3.1. Utilizando-se do mesmo, gerou-se uma população inicial de moléculas, com o tamanho especificado por *PopSize*.

Figura 39 – Classe molécula para a p-mediana não-capacitada

```
"Molecule" class
class Molecule
Attributes: ω, PE, KE, NumHit, MinStruct, MinPE, MinHit
Method:
Molecule() \\constructor
    {
    Randomly generate ω in the solution space
    PE ← f (ω)
    KE ← InitialKE
    NumHit ← 0 and MinStruct ← ω and MinPE ← PE and MinHit ← 0
    }
OnwallIneffectiveCollision()
Decomposition()
IntermolecularIneffectiveCollision()
Synthesis()
end class
```

Fonte: Lam e Li (2012)

5 COLISÕES

Uma molécula pode atingir uma parede do recipiente ou colidir com uma outra molécula. Isto é decidido gerando-se um número aleatório *b* entre [0, 1]. Se $b > MoleColl$ ou se o sistema tiver apenas uma molécula, uma colisão unimolecular ocorrerá. Caso contrário, segue-se uma colisão intermolecular. Para uma colisão unimolecular, uma molécula da população é selecionada aleatoriamente e, então, decide-se se ocorrerá com ela uma colisão ineficaz com a parede ou uma decomposição, de acordo com o critério de decomposição escolhido. O critério de decomposição foi definido como:

$$NumHit - MinHit > \alpha$$

De forma análoga, para uma colisão intermolecular, selecionam-se aleatoriamente duas moléculas da população e, então, determina-se se haverá uma colisão ineficaz intermolecular ou uma síntese verificando-se o critério de síntese para as moléculas escolhidas. A síntese ocorrerá se todas as moléculas atenderem o seguinte critério:

$$KE \leq \beta$$

As desigualdades acima controlam o grau de diversificação através dos parâmetros α e β. Valores adequados de α e β promovem um equilíbrio adequado entre diversificação e intensificação.

Após a ocorrência de uma reação elementar, deve-se verificar se a condição de conservação de energia é obedecida. Caso isso não haja ocorrido a mudança deve ser descartada. A seguir, é verificada se qualquer uma das soluções produzidas naquela operação possui um valor de função objetivo mais baixo do que a melhor solução obtida até o momento. Se assim o for, registra-se tal solução como a melhor até o momento.

Se nenhum critério de parada for atingido, inicia-se uma nova iteração, após armazenar-se a melhor solução dentre as melhores soluções de toda a população, ou seja, após armazenar-se o *MinStruct* com o menor *MinPE* entre todas as moléculas.

6 FINALIZAÇÃO DO ALGORITMO

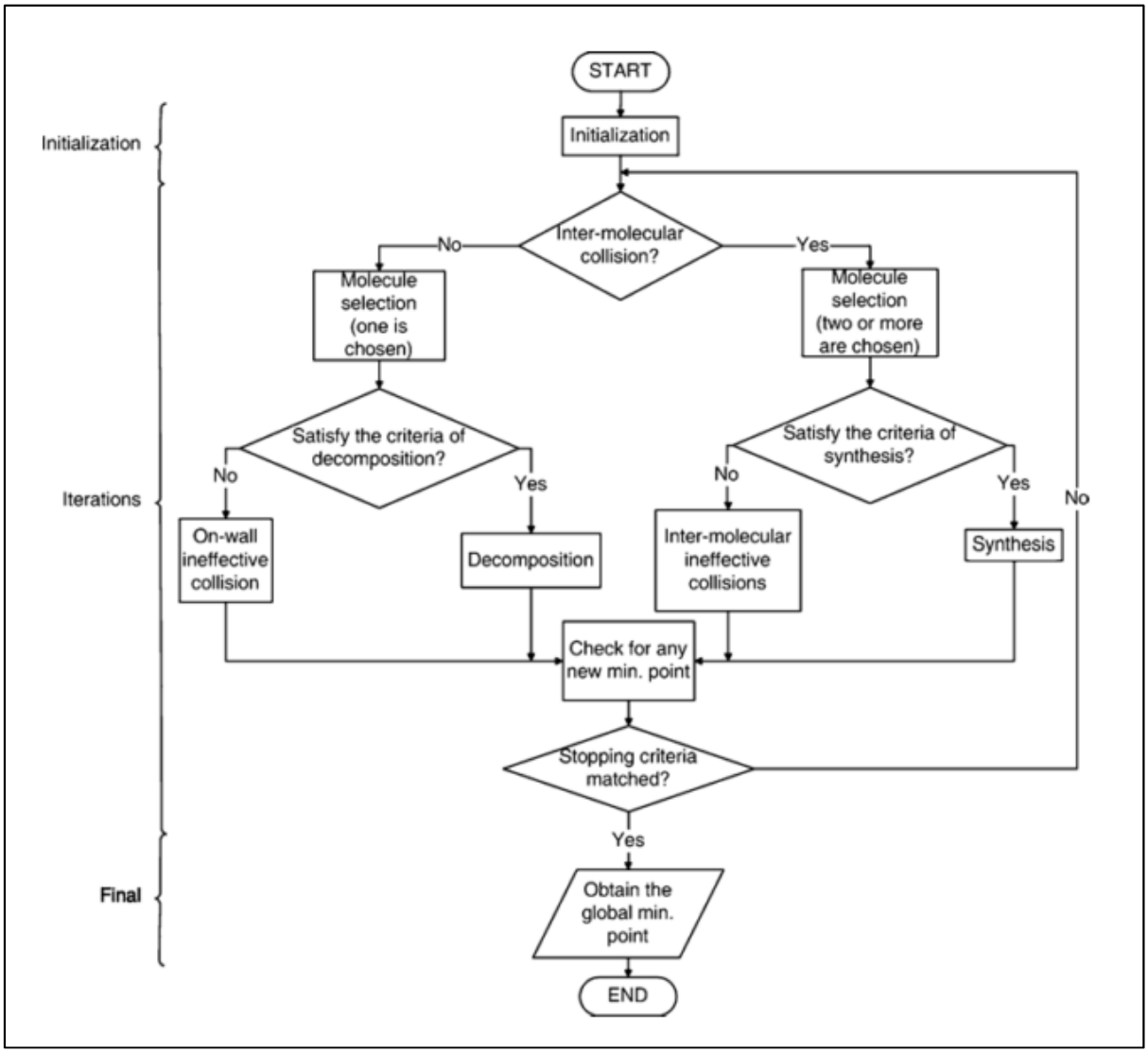
Se algum dos critérios de parada for atingido, segue-se para a fase final. Os critérios de parada utilizados foram o número máximo de iterações e o número máximo de iterações realizadas sem melhorias, ou seja, sem que o menor *MinPE* entre todas as moléculas tenha mudado. Na fase final, a melhor solução encontrada é exibida e o algoritmo, finalizado. O pseudocódigo principal do CRO é mostrado na Figura 40.

Figura 40 – Pseudocódigo principal do CRO

```
Implementation of CRO algorithm
Input: Objective function f and the parameter values
\\ Initialization
Set PopSize, KELossRate, MoleColl, buffer, InitialKE, α and β
Create PopSize number of molecules
\\ Iterations
while the stopping criteria not met do
    Generate b in [0, 1]
    if b > MoleColl then
        Randomly select one molecule Mω
        if Decomposition criterion met then
            Trigger Decomposition
        else
            Trigger OnwallIneffectiveCollision
        end if
    else
        Randomly select two molecules Mω1 and Mω2
        if Synthesis criterion met then
            Trigger Synthesis
        else
            Trigger IntermolecularIneffectiveCollision
        end if
    end if
    Check for any new minimum solution
end while
\\ The final stage
Output the best solution found and its objective function value
```

Fonte: Lam e Li (2012)

Figura 41 – Diagrama esquemático do CRO



Fonte: Lam e Li (2012)

7 OPERADORES

A seguir estão descritos os operadores usados nas transformações ineficazes uni e multimoleculares, nas sínteses e nas decomposições:

7.1 Operador Fast Interchange

Nas colisões ineficazes uni e multimoleculares um operador baseado no algoritmo *Fast Interchange* proposto por Whitaker (1983) é utilizado. No trabalho de Whitaker, três ingredientes eficientes são incorporados na heurística de troca padrão:

- Avaliação de movimento (*Move evaluation*), onde a melhor estação a ser removida é encontrada quando a estação a ser adicionada é conhecida. O pseudocódigo do procedimento *Move* é apresentado na Figura 42;

- Atualização da primeira e da segunda estação mais próxima de cada centro de consumo (*Update*). O pseudocódigo do procedimento *Update* é apresentado na Figura 43;

- Estratégia restrita a primeira melhoria, onde cada estação é considerada para ser adicionada apenas uma vez. Hansen e Mladenovic (1997) implementaram uma versão modificada do algoritmo *Fast Interchange* para ser usada na metaheurística *Variable Neighborhood Search* (VNS), na qual o mesmo foi modificado para uma estratégia de obtenção da maior melhoria (*best improvement*). Na presente implementação, o algoritmo modificado por Hansen e Mladenovic (1997) foi utilizado, sendo também possível configurá-lo, através de um parâmetro de entrada, para terminar assim que a primeira melhoria houver sido obtida, ou executá-lo até que a melhor melhoria tenha sido alcançada. O pseudocódigo do algoritmo *Fast Interchange* modificado é apresentado na Figura 44.

Figura 42 – Procedimento *Move* do algoritmo *Fast Interchange*

Move evaluation: a change w in the objective function value is evaluated when the facility (service station) that is added (denoted with goin) to the current solution is known, while the best one to be dropped (denoted with goout) is to be found. In the description of the heuristic, we use the following notation:

c1(i) - median(closest facility) of user i, i = 1,...,n;

c2(i) - second closest facility of user i, i = 1,..., n;

d(i,j) - distance between user i and facility j, i = 1,...,n; j = 1,...,m;

goin - index of inserted facility (input value);

w - change in the objective function value obtained by the best interchange;

$x_{cur}(i)$, *i = 1,...,p - current solution (indices of medians);*

v(j) - change in the objective function value obtained by deleting each facility currently in the solution, (j (j = $x_{cur}(i)$, *i = 1,...,p);*

goout - index of deleted facility (output value).

Algorithm Move (cl,c2,d,x_{cur}*,goin,n,p,w,goout):*

Initialization:

```
w = 0
v(x_cur(i) = 0), for all i = 1,...,p.
Best deletion:
For each user i (i = 1,...,n) do:
  If d(i,goin) < d(i,c1(i))
    w = w + d(i,goin) - d(i,cl(i)
  else
    // update the change in value obtained by deleting cl(i):
    v(c1(i))= v(c1(i))+ min(d(i,goin),d(i,c2(i)))-d(i,cl(i))
  End if
End For i
Find g = min(v(x_cur(l)),l = 1,..., p) and facility goout [index x_cur(l)) where this minimum is reached].
w = w + g
```

Fonte: Hansen e Mladenovic (1997)

Figura 43 – Procedimento *Update* do algoritmo *Fast Interchange*

```
Updating first and second closest facilities. In the Move evaluation procedure, both the closest [cl(i)] and
second closest facility [c2(i)] for each user i must be known in advance. Among formal variables in the
description of algorithm Update that follows, arrays c1(i) and c2(i), i=1,... ,n are both input and output
variables. All other formal variables in the list are input only.

Algorithm Update (d,goin,goout,n,p,cl,c2)
   For each user i (i = 1 to n) do:
      // For users whose center is deleted, find new one
      If cl(i) = goout then
         If d(i,goin) <d(i,c2(i)) then
            cl(i) = goin
         else
            cl(i) = c2(i)
            // Find second closest facility for user i
            Find center l* where d(i,l) is minimum (for l = 1,...,p, l <> cl(i));
            c2(i) = l*
         End if
      else
         If d(i,cl(i)) > d(i,goin) then
            c2(i)= cl(i)
            cl(i)= goin
         else
            If d(i,goin) < d(i,c2(i)) then
               c2(i) = goin
            else
               If c2(i) = goout then
                  Find center l* where d(i,l) is minimum (for l = 1,...,p, l <> cl(i));
                  c2(i)= l*
               End if
            End if
         End if
   End for i
```

Fonte: Hansen e Mladenovic (1997)

Figura 44 – Pseudocódigo do algoritmo *Fast Interchange*

Fast interchange heuristic, which uses procedures *Move* and *Update*, described before, is as follows:

Initialization

Denote a random permutation of facilities {1,...,m} with $x_{opt}(j)$, j = 1,...,m. Let the first p of them represent an initial solution; find the corresponding objective function value f_{opt}; find the closest and second closest facility for each user i,

i.e., find arrays cl(i), cl(i), c2(i), i=l ,...,n;

Iteration step:

$w^* = \infty$

For goin = $x_{opt}(p + 1)$ to $x_{opt}(m)$

// Add facility goin not in the solution into it and

// find the best deletion

Run procedure Move(cl,c2,d,x_{cur},goin,n,p,w,goout);

// Keep the best pair of facilities to be interchanged

If w < w then*

w = w,*

goin = goin*

goout = goout*

End if

End For goin

Termination:

// If no improvement in the neighborhood, Stop

If w > 0 then Stop*

Updating:

// Update objective function value:

$f_{opt} = f_{opt} + w^*$

// Update x_{opt}

Interchange position of posição x_{opt}(goout) with x_{opt}(goin*)*

// Update closest and second closest facilities:

Update(d,goin,goout*,n,p,cl,c2)*

Return to Iteration step

Fonte: Hansen e Mladenovic (1997)

7.2 Operador Half-total change

É o operador utilizado para efetuar a decomposição de uma molécula em duas outras moléculas, conforme descrito no item 2. Como o seu nome implica, uma nova solução é produzida a partir de uma já existente, mantendo-se metade dos valores existentes e atribuindo-se novos valores à metade restante.

Suponha que se tente produzir duas novas soluções $\omega'_1 = [\omega'_1(i), 1 \leq i \leq n]$ e $\omega'_2 = [\omega'_2(i), 1 \leq i \leq n]$ *a partir de* $\omega = [\omega(i), 1 \leq i \leq n]$. Para obter-se ω'_1, primeiro copia-se ω para ω'_1 e, em seguida, escolhe-se aleatoriamente $\lfloor N/2 \rfloor$ elementos no vetor ω'_1, onde $\lfloor . \rfloor$ retorna o maior inteiro igual ou menor que o argumento. Para cada um destes elementos, por exemplo $\omega'_1(i)$, atribui-se um novo valor de acordo com as restrições do problema. Como os elementos são escolhidos aleatoriamente, ω'_1 e ω'_2 são bastante diferentes entre si, e também de ω'. Na presente implementação, o operador *Half-total change* assegura que os elementos de ω' estejam presentes em ω'_1 ou ω'_2, mas não em ambos. Isto garante que todas as estações pertencentes a ω' estejam presentes, seja em ω'_1 ou ω'_2.

7.3 Operador Distance Preserving Crossover (DPX)

É o operador usado para efetuar a síntese de duas moléculas em uma única molécula, conforme descrito no item 2. O operador DPX foi utilizado por Merz e Freisleben (1997) para resolução do problema de assinalamento quadrático, através de Algoritmo Genético, se mostrando bem adaptado ao problema da p-mediana, uma vez que se baseia unicamente na noção de distância entre soluções. Sejam π_1 e π_2 soluções válidas para um dado problema. A distância T, entre soluções é definida como:

$$\boldsymbol{T(\pi_1, \pi_2)} \quad = |\{\boldsymbol{i} \in \{\mathbf{1}, \ldots, \boldsymbol{n}\} | \, \boldsymbol{\pi_1(i)} \neq \boldsymbol{\pi_2}\,(\boldsymbol{i})\}|$$

Tal como definido acima, T representa quantidade de estações presentes em π_1 que não se se encontram em π_2. O DPX tem como objetivo produzir um descendente que possua a mesma distância de cada um de seus pais, sendo esta distância igual à distância entre os próprios pais. Sejam dois pais A e B, contendo soluções viáveis. Primeiro, todas as atribuições de estações contidas em ambos os pais são copiadas para o descendente C. As posições restantes são então aleatoriamente preenchidas com estações ainda não atribuídas, assegurando-se que nenhuma atribuição que possa ser encontrada em apenas um dos pais seja inserida no descendente. Desta

forma, obtém-se um descendente *C*, para qual a condição *T(C,A) = T(C,B) = T(A,B)* se mantém. Tal cruzamento é altamente disruptivo, forçando as subsequentes buscas locais a explorarem uma região diferente do espaço de solução, onde melhores soluções poderão ser encontradas.

8 METODOLOGIA

A metaheurística CRO para a resolução do problema da p-mediana foi implementada na forma de uma aplicação codificada na linguagem de programação *Microsoft Visual C# 2015.*

A base de dados *OR-Library* (BEASLEY, 1990), amplamente utilizada pela comunidade científica para testes e validação de problemas de p-mediana, foi utilizada para avaliar a qualidade e desempenho da presente implementação. A base contém 40 problemas do tipo p-mediana, com tamanhos entre 100 e 900 estações/consumidores e 5 a 100 medianas.

Inicialmente, cada um dos 40 problemas da *OR-Library* foi resolvido, de forma exata, utilizando o otimizador IBM CPLEX (INTERNATIONAL BUSINESS MACHINES CORPORATION, 2017) vs. 12.6. A seguir, na mesma plataforma computacional, cada problema foi resolvido 20 vezes pelo aplicativo CRO, utilizando-se a mesma configuração para todos os problemas. A configuração dos parâmetros operacionais do CRO utilizadas nos problemas foi a seguinte: ***PopSize = 10, KELossRate = 0,8, MoleColl = 0,2, InitialKE = 100.000, α = 1, β = 10.000 e buffer = 0***. Como critério de parada, limitou-se o número máximo de iterações em 5.000 e o número máximo de iterações realizadas sem melhorias em 500 iterações.

Durante os testes, observou-se que os melhores resultados, em termos de desempenho, foram obtidos com populações iniciais pequenas, entre 10 e 100 moléculas. Isto permitiu que ocorresse intensificação suficiente, ou seja, que existissem colisões unimoleculares e multimoleculares ineficazes em quantidade tal que se pudesse alcançar os mínimos locais entre as vizinhanças, sem que o número de iterações aumentasse demasiadamente.

9 RESULTADOS

Os resultados computacionais para a p-mediana clássica são apresentados na seção 4.

APÊNDICE C – Implementação da metaheurística CRO em linguagem C#

1 APLICAÇÃO

A metaheurística CRO para a resolução do problema proposto foi implementada na forma de uma aplicação de linha de comando para execução em ambiente operacional Microsoft Windows. Devido às características da metaheurística CRO, a linguagem de programação orientada a objeto Microsoft Visual C# 2015 foi utilizada. A característica de orientação a objetos da linguagem de programação C# permitirá modelar-se com maior exatidão os componentes básicos do CRO, como as moléculas, com suas propriedades e reações químicas.

O código foi desenvolvido utilizando ambiente de desenvolvimento Microsoft Visual Studio 2015 Community Edition, versão 14.0.25123.00 Update 2, com Microsoft .NET Framework 4.6.01055.

A plataforma de hardware se constitui de um computador pessoal (PC), com processador Intel Core i7 de oito núcleos de 2.3 GHz, 16GB de memória RAM, executando sistema operacional Windows, versão 10.0. A quantidade de memória reservada para a aplicação deve ser de, pelo menos, 512 MB.

As moléculas foram implementadas na forma instâncias de classes. Cada instância possui um conjunto de propriedades que correspondem aos atributos definidos no CRO a serem armazenados para cada molécula. O código parcial, em linguagem de programação C#, que implementa a classe molécula encontra-se na Figura 46. As reações elementares foram implementadas na forma de métodos, os quais fazem parte de uma classe que implementa a metaheurística CRO. Os demais algoritmos necessários à implementação dos operadores de intensificação e diversificação, também foram implementados na forma de classes. Os elementos em comum a todos os algoritmos farão parte de uma classe abstrata, da qual todas as outras classes que implementam os diversos algoritmos do CRO são derivadas. Um diagrama de classes simplificado da aplicação pode ser visto na Figura 47.

2 ARQUIVOS COMPONENTES DA SOLUÇÃO

A pasta contendo a solução implementada é denominada de **"CRO**", e se encontra na raiz do sistema de arquivos da mídia de distribuição. Dentro desta pasta encontram-se diversas outras pastas e arquivos, contendo o arquivo de definição da solução, arquivo de definição do projeto principal, arquivos contendo os códigos em C# de definição e implementação das diversas

classes, arquivo executável compilado e arquivos de entrada e saída. A seguir estão listados os principais arquivos componentes da presente implementação, organizados por pasta, tendo como referência a pasta **\CRO**:

- **\CRO\CRO.sln:** contém a definição da solução, conforme requerido pelo ambiente Microsoft Visual Studio 2015;
- **\ChargingStation:** Contém os demais arquivos e pastas do projeto:
  - **CRO.csproj:** definições do projeto (requerido pelo Visual Studio 2015);
  - **CRO.csproj.user:** configurações do ambiente de usuário (requerido pelo Visual Studio 2015);
  - **HeuristicAlg.cs:** Definição dos métodos e propriedades comuns a todos os algoritmos de otimização utilizados no projeto;
  - **CROAlg.cs:** Implementação dos métodos e propriedades do CRO;
  - **DPXAlg.cs:** Implementação do operador DPX;
  - **FastInterchangeAlg.cs:** Implementação do operador *Fast Interchange*;
  - **HalfTotalChange.cs:** Implementação do operador *Half-total change*;
  - **Molecule.cs:** Implementação da classe Molécula;
  - **Neighborhood.cs: :** Implementação do operador *Neighborhood Search;*
  - **OPLClasses.cs:** Implementação das classes e métodos para escrita de arquivos em disco no formato aceito pelo otimizador CPLEX;
  - ***OR-Library*pcapClasses.cs:** Implementação das classes e métodos para a leitura da biblioteca da *OR-Library* contendo os problemas da p-mediana capacitada.
  - ***OR-Library*pmedClasses.cs:** Implementação das classes e métodos para a leitura da biblioteca da *OR-Library* contendo os problemas da p-mediana não-capacitada.
  - **ProblemData.cs:** Definição da estrutura do problema a ser resolvido;
  - **Program.cs:** Programa principal, que contém o método ***main();***
  - **ProgramCommonMethods.cs:** Coletânea de métodos estáticos para implementação de diversas funcionalidades secundárias, como exibição de vetores e matrizes na tela, cópia de objetos, e outros;
  - **ProgramCROPcap.cs:** Contém a chamada ao ponto de entrada do algoritmo CRO, para os problemas da p-mediana capacitada;
  - **ProgramCROPmed.cs:** Contém a chamada ao ponto de entrada do algoritmo CRO, para os problemas da p-mediana não-capacitada;

- **Solution.cs:** Contém a definição da solução do problema, ou seja, as varáveis de decisão *x* e *y,* assim como o valor da função objetivo correspondente;
- **\bin\Release:** Pasta que contém o programa executável, os arquivos de entrada e o arquivo de saída:

- **CRO.exe:** Programa executável (código compilado);
- **pmed*xx*.txt:** Série de arquivos texto contendo os problemas da biblioteca *OR-Library* referentes a p-mediana não-capacitada. São, ao todo, 40 arquivos, numerados de 1 a 40. Exemplo: O problema 15 está contido em p-med15.txt;
- **p-medcap.txt:** Arquivo texto contendo todos os problemas da biblioteca *OR-Library* referentes a p-mediana capacitada;
- **pmedout.txt:** Arquivo contendo o resultado do processamento, ou seja, a solução para um ou mais problemas da biblioteca *OR-Library*.

3 ENTRADA DE DADOS

A entrada e saída é feita através de arquivos em disco, armazenados na mesma pasta onde se encontra o programa executável. Os arquivos de entrada devem ser os arquivos originais contidos na biblioteca *OR-Library* (BEASLEY, 1990) para os problemas da p-mediana capacitada ou não-capacitada. Todas as transformações de dados necessárias a criação das matrizes de energia, capacidade e distância são feitas internamente pela aplicação, não sendo requerida nenhuma modificação nos arquivos originais. Uma descrição detalhada do formato interno dos arquivos está pode ser encontrada na etapa 3.4.

Outros problemas da p-mediana, que não os da biblioteca *OR-Library* (BEASLEY, 1990) poderão ser resolvidos, desde que o arquivo de entrada seja compatível, em formato, com os da referida biblioteca. Futuras versões do aplicativo darão suporte à outras bibliotecas e formatos de entrada, incluindo àquelas contendo dados obtidos de sistemas de navegação.

Os arquivos em disco a serem processados são especificados através de um parâmetro da linha de comando de chamada do aplicativo.

4 SAÍDA DE DADOS

Os dados de saída, contendo a solução dos problemas, são gravados em um arquivo em disco. O nome do arquivo de saída deverá ser especificado através de um parâmetro da linha de comando.

O arquivo de saída é do tipo **CSV** (***Comma Separated Values***) e seu formato interno é descrito a seguir:

A primeira linha do arquivo de saída contém o nome das colunas armazenadas no arquivo de saída, separadas por vírgula:

- O tipo de algoritmo utilizado (**Alg**);
- O nome do arquivo de entrada (**ProblemID**);
- O número de estações de carregamento encontradas na solução (**P**);
- O número de estações de carregamento candidatas (**NumStations**);
- O número de centros de consumo existentes (**NumClusters**);
- O valor final da função objetivo (**Objective**);
- O tempo de execução em *ms* (**Duration**);
- Uma relação de colunas (**ST1, ST2, ST3...)** em quantidade suficiente para registrar a maior solução possível dentre todos os problemas submetidos à execução.

O arquivo de saída conterá, da segunda linha em diante, as soluções encontradas para os diversos problemas submetidos à execução, conforme ilustrado na Figura 45. Por exemplo, a linha "**CRO,pmed01,5,100,100,5819,612,64,98,12,90,6**", indica que para o problema armazenado em pmed01.txt, composto de 100 estações de carregamento candidatas e 100 centros de consumo, ao final do processamento, a metaheurística CRO selecionou 5 estações de carregamento dentre as 100 candidatas, o valor final da função objetivo foi de 5819 e as estações selecionadas foram as 612,64,98,12,90 e 6.

Figura 45 – Arquivo de saída do aplicativo CRO

```
Alg,ProblemID,P,NumStations,NumClusters,Objective,Duration,ST0,ST1,ST2,…
CRO,pmed01,5,100,100,5819,612,64,98,12,90,6
CRO,pmed01,5,100,100,5819,499,6,90,98,64,12
CRO,pmed01,5,100,100,5819,442,12,90,64,6,98
CRO,pmed01,5,100,100,5819,466,90,6,98,12,64
CRO,pmed01,5,100,100,5819,436,64,12,90,6,98
CRO,pmed01,5,100,100,5819,453,90,64,12,98,6
CRO,pmed01,5,100,100,5819,436,64,98,6,90,12
CRO,pmed01,5,100,100,5819,439,12,98,6,90,64
CRO,pmed01,5,100,100,5819,439,90,98,6,12,64
CRO,pmed01,5,100,100,5819,445,6,98,64,12,90
CRO,pmed02,10,100,100,4093,505,94,44,36,5,11,98,7,66,40,57
CRO,pmed02,10,100,100,4093,506,66,90,94,36,5,98,7,11,40,44
CRO,pmed02,10,100,100,4093,518,5,44,90,98,40,66,11,7,94,36
CRO,pmed02,10,100,100,4093,591,44,94,98,66,40,90,5,11,7,36
CRO,pmed02,10,100,100,4093,494,11,7,5,36,98,94,40,90,44,66
CRO,pmed02,10,100,100,4093,490,11,98,44,5,40,36,90,94,7,66
CRO,pmed02,10,100,100,4093,482,94,66,90,40,98,11,7,5,44,36
CRO,pmed02,10,100,100,4093,492,66,40,94,90,11,98,5,44,36,7
```

Fonte: Elaborado pelo autor (2017)

5 DIAGRAMA DE CLASSES

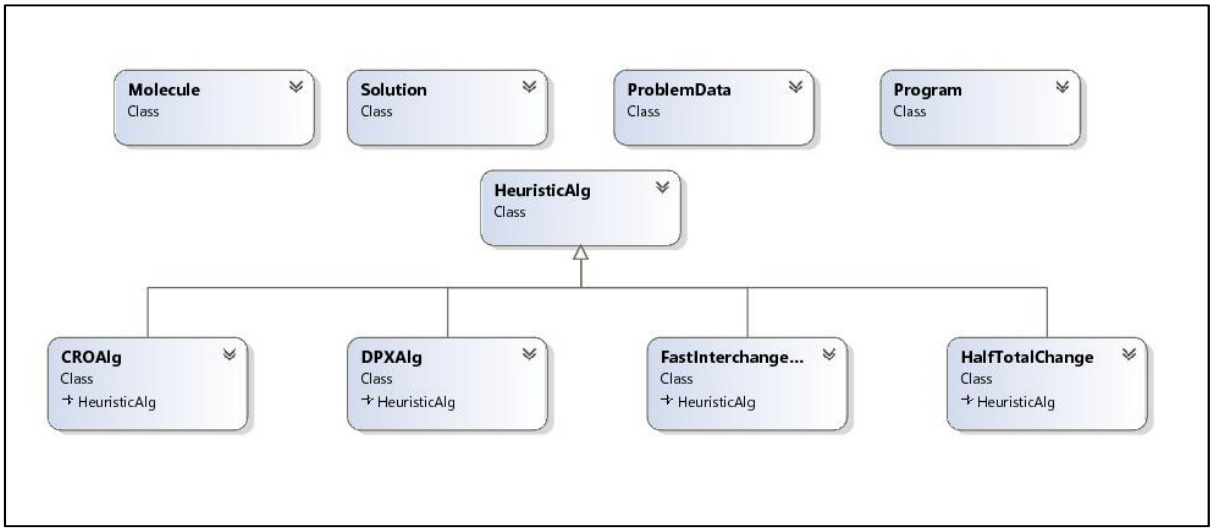
Um diagrama contendo as principais classes da aplicação é mostrado na Figura 47. A classe *Molecule* implementa a estrutura da molécula, bem como o seu construtor de classe, como definido pelo CRO (LAM; LI, 2012). O código parcial da classe molécula é mostrado na Figura 46. A classe *Solution* contém as variáveis de decisão *x* e *y,* bem como o valor da função objetivo correspondente. A classe *ProblemData* contém a matriz de energia *d,* o vetor contendo o número de veículos trafegando a um centro de consumo, $n^{(ev)}$, além do número de estações de carregamento candidatas, o número de centros de demanda, e o número de medianas. A classe *HeuristicAlg* contém os métodos e propriedades comuns a todos os algoritmos de otimização implementados na solução. A classe *CROAlg* contém os métodos e propriedades específicos da metaheurística CRO, incluindo o código principal, e os das colisões ineficazes uni e intermoleculares, da decomposição e da síntese. A classe *FastInterchangeAlg* contém os métodos e propriedades específicos da heurística *Fast Interchange*, incluindo o código principal e os métodos que implementam os procedimentos *Move* e *Update*. A classe *NeighborhoodAlg* contém os métodos e propriedades específicos da heurística *Neighbodhood Search.* A classe *HalfTotalChangeAlg* contém os métodos e propriedades específicos da heurística *Half-total change*. Por fim, a classe *DPXAlg* contém os métodos e propriedades específicos da heurística *DPX*.

Figura 46 – Implementação da classe Molécula em C#

```
class Molecule
{
   // Molecule ID
   public int MolId { get; set; }

   // Solution that a molecule carries
   public Solution W { get; set; }

   // Molecule's potential energy
   public double PE { get; set; }

   // Molecule's knectic energy
   public double KE { get; set; }

   // Number of hits
   public int NumHit { get; set; }

   // Min struct
   public Solution MinStruct { get; set; }

   // Min PE
   public double MinPE { get; set; }

   // Min hit
   public double MinHit { get; set; }

   // Problem's Data
   public ProblemData Problem { get; set; }

   // Constructor
   public Molecule (int molId, Solution w , ProblemData problem, double initialKE)
   {
      W = Program.Clone<Solution> (w);

      // Problem's data
      Problem = problem;
      ...
```

Fonte: Elaborado pelo autor (2017)

Figura 47 – Diagrama de classes da aplicação CRO em C#



Fonte: Elaborado pelo autor (2017)

6 EXECUÇÃO DO APLICATIVO

Sintaxe da linha de comando:

CRO <p1> <p2> <p3> <p4> <p5> <p6>

Parâmetros:
<p1> : Heurística a ser aplicada. Os valores aceitos são: "*cropmed*" para problemas da p-mediana não-capacitada, "*cropcap*" para problemas da p-mediana capacitada, "*cropmedbaouche*" para problemas da p-mediana seja capacitada ou não-capacitada usando o modelo proposto no item 3.1.

<p2> : Nome do arquivo de entrada. Podem ser usados caracteres coringa "*" e "?" para referenciar múltiplos arquivos, na mesma pasta. Caso uma rota completa não seja fornecida, a pasta corrente será usada.

<p3> : Fator multiplicador da função objetivo, representando o custo do Kilowatt-hora. Para que os resultados da resolução da biblioteca *OR-Library* (BEASLEY, 1990), para os problemas da p-mediana capacitada e não-capacitada, possam ser comparados com a literatura, o valor deve ser sempre "1";

<p4> : Distância mínima entre estações de carregamento. Para que os resultados da resolução da biblioteca *OR-Library* (BEASLEY, 1990), para os problemas da p-mediana capacitada e não-capacitada, possam ser comparados com a literatura, o valor deve ser sempre "0";

<p5> : Nome do arquivo de saída. Caso uma rota completa não seja fornecida, a pasta corrente será usada.

<p6> : Número do problema a ser resolvido. Só deve ser informado se o problema for da p-mediana capacitada, já que todos os problemas se encontram em um só arquivo. Se "*" for usado, todos os problemas serão resolvidos.

Exemplos:

- Resolver o primeiro problema da p-mediana não-capacitada da biblioteca *OR-Library*:
  CRO *"cropmed" "pmed01.txt" "1" "0" "pmedout.txt"*
- Resolver todos os problemas da p-mediana não-capacitada da biblioteca *OR-Library*, presentes na pasta que contém o programa executável:
  CRO *"cropmed" "pmed*.txt" "1" "0" "pmedout.txt"*
- Resolver todos os problemas da p-mediana capacitada da biblioteca *OR-Library*, presentes no arquivo *p-medcap.txt,* na pasta que contém o programa executável:
  CRO *"cropcap" " p-medcap.txt" "1" "0" "pmedout.txt" "*"*

A Figura 48 mostra a execução do aplicativo para a primeiro problema da p-mediana não-capacitada da biblioteca *OR-Library*.

Figura 48 – Resolução do primeiro problema da *OR-Libraryt*

```
C:\IFES\MTS\OTM\Trabalho\CRO\CRO\bin\Release>CRO "cropmed" "pmed01.txt" "1" "0" "pmedout.txt"
Running CRO optimization...
----------------------------------------------------------
Library ID: pmed01
Input file: C:\IFES\MTS\OTM\Trabalho\CRO\CRO\bin\Release\pmed01.txt
Loading input file...
ProblemData.Alpha = 1
ProblemData.R = 0
ProblemData.NumClusters = 100
ProblemData.NumStations = 100
ProblemData.P = 5
**********************************************************************************************************************
**********
Running CRO algorithm...
**********************************************************************************************************************
**********
PopSize = 10
KELossRate = 0.8
MoleColl = 0.2
InitialKE = 10000
Alpha =1
Beta = 2000
maxIterations = 5000
maxIterationsWithSameObjective = 500
minMol = 1
maxMol = 50
----------------------------------------------------------------------------------------------------------------------
Mol no. = 0   Stations  = 5   W.Obj = 8783   PE = 8783   KE = 10000   NumHit = 0   MinPE  = 8783   MinHit = 0   TotalCost = 8783
Clusters = 100
Stations = 95 37 66 12 68
----------------------------------------------------------------------------------------------------------------------
Mol no. = 1   Stations  = 5   W.Obj = 7491   PE = 7491   KE = 10000   NumHit = 0   MinPE  = 7491   MinHit = 0   TotalCost = 7491
Clusters = 100
Stations = 16 2 22 74 55
----------------------------------------------------------------------------------------------------------------------
Mol no. = 2   Stations  = 5   W.Obj = 7491   PE = 7491   KE = 10000   NumHit = 0   MinPE  = 7491   MinHit = 0   TotalCost = 7491
Clusters = 100
Stations = 16 2 22 74 55
----------------------------------------------------------------------------------------------------------------------
Mol no. = 3   Stations  = 5   W.Obj = 7717   PE = 7717   KE = 10000   NumHit = 0   MinPE  = 7717   MinHit = 0   TotalCost = 7717
Clusters = 100
Stations = 58 80 62 97 63
----------------------------------------------------------------------------------------------------------------------
Mol no. = 4   Stations  = 5   W.Obj = 7717   PE = 7717   KE = 10000   NumHit = 0   MinPE  = 7717   MinHit = 0   TotalCost = 7717
Clusters = 100
Stations = 58 80 62 97 63
----------------------------------------------------------------------------------------------------------------------
Mol no. = 5   Stations  = 5   W.Obj = 7717   PE = 7717   KE = 10000   NumHit = 0   MinPE  = 7717   MinHit = 0   TotalCost = 7717
Clusters = 100
Stations = 58 80 62 97 63
----------------------------------------------------------------------------------------------------------------------
…
----------------------------------------------------------------------------------------------------------------------
Mol no. = 9   Stations  = 5   W.Obj = 7717   PE = 7717   KE = 10000   NumHit = 0   MinPE  = 7717   MinHit = 0   TotalCost = 7717
Clusters = 100
Stations = 58 80 62 97 63
----------------------------------------------------------------------------------------------------------------------
**********************************************************************************************************************
**********
Objective final optimized value = 5819
Execution time: 0.534s
Number of medians = 5
----------------------------------------------------------------------
Finished...
Pressione qualquer tecla para continuar...
```

Fonte: Elaborado pelo autor (2017)