Universidade Estadual de Campinas
Faculdade de Tecnologia

Felipe Favaro Müller

Algoritmos Genéticos Aplicados Ao Problema De Roteamento De Veículos

Limeira
2019

Felipe Favaro Müller

Algoritmos Genéticos Aplicados Ao Problema De Roteamento De Veículos

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Tecnologia, na área de Sistemas de Informação e Comunicação.

Orientador: Prof. Dr. Luis Augusto Angelotti Meira

Este exemplar corresponde à versão final da Dissertação defendida por Felipe Favaro Müller e orientada pelo Prof. Dr. Luis Augusto Angelotti Meira.

Limeira
2019

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.
ORCID: https://orcid.org/0000-0002-3572-336X

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

M913a

Müller, Felipe Fávaro, 1993-
Algoritmos genéticos aplicados ao problema de roteamento de veículos / Felipe Fávaro Müller. – Limeira, SP : [s.n.], 2019.

Orientador: Luis Augusto Angelotti Meira.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Problema de roteamento de veículos. 2. Algoritmos genéticos. 3. Otimização combinatória. I. Meira, Luis Augusto Algelotti, 1979-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Genetic algorithms applied to the vehicle routing problem
Palavras-chave em inglês:
Vehicle routing problem
Genetic algorithms
Combinatorial optimization
Área de concentração: Sistemas de Informação e Comunicação
Titulação: Mestre em Tecnologia
Banca examinadora:
Luis Augusto Angelotti Meira [Orientador]
Flávio Keidi Miyazawa
Marco Antonio Garcia de Carvalho
Data de defesa: 31-01-2019
Programa de Pós-Graduação: Tecnologia

FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Mestre em Tecnologia na área de concentração de Sistemas de Informação e Comunicação, a que submeteu o aluno Felipe Favaro Müller, em 31 de Janeiro de 2019 na Faculdade de Tecnologia- FT/ UNICAMP, em Limeira/SP.

Prof. Dr. Luis Augusto Angelotti Meira

Presidente da Comissão Julgadora

Prof. Dr. Flávio Keidi Miyazawa

IC/UNICAMP

Prof. Dr. Marco Antonio Garcia de Carvalho

FT/UNICAMP

Ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós Graduação da FT.

Without commitment, you'll never start,
but more importantly,
without consistency, you'll never finish.
(Denzel Washington)

Agradecimentos

Primeiramente eu gostaria de agradecer a meu orientador, Prof. Dr. Luis Augusto Angelotti Meira, FT - Unicamp, que sempre manteve uma agenda flexível para nossas reuniões e foi bastante prestativo, mesmo antes deste mestrado iniciar oficialmente.

Também gostaria de agradecer ao meu colega e funcionário de T.I. na FT, Bruno Luciano Amadio Caires, que foi de grande ajuda no uso de recursos do Laboratório de Simulação e de Computação de Alto Desempenho (LaSCADo) para a conclusão deste trabalho.

Finalmente, sou extremamente grato a meus familiares e amigos pelo apoio que recebi ao longo deste período.

Resumo

O Problema de Roteamento de Veículos (do inglês *Vehicle Routing Problem*, VRP) é uma generalização do clássico Problema do Caixeiro Viajante (do inglês *Traveling Salesman Problem*, TSP) e pode ser definido de forma ampla como uma classe de problemas que envolvem visitar "clientes" com "veículos". Tal problema surgiu em 1959 e possui uma estrutura desafiadora a ser otimizada. O VRP modela diversas situações reais, como o roteamento de veículos para atender chamadas ou ocorrências, definição de rotas para carteiros, definição de rotas para coleta de lixo, definição de itinerários de ônibus fretados e vans, entre muitas outras. Este trabalho desenvolveu algoritmos genéticos combinados à busca local e os aplicou a duas variantes do problema, focando primariamente na minimização do número de veículos. A primeira variante modela um caso real baseado nos correios de Artur Nogueira, SP, chamada PostVRP. Trata-se de um *benchmark* de grande escala, com instâncias de até 30.000 pontos de entrega, cujos limitantes são desconhecidos. A segunda variante é chamada CVRP (do inglês *Capacitated Vehicle Routing Problem*) e o conjunto de instâncias aqui utilizado já possui ótimos conhecidos para fins de comparação. Foram feitos experimentos com quatro variações de algoritmo genético: usando *Best Cost Route Crossover* (BCR) ou *Order Based Crossover* (OX) e otimizando com e sem busca local. Os algoritmos propostos obtiveram os ótimos resultados conhecidos em 46% dos casos para as instâncias CVRP, e ficaram mais de 10% piores em apenas 9%. O uso de busca local combinado ao algoritmo genético acelerou a convergência das soluções e, entre os dois operadores de *crossover* testados, o operador BCR se sobressaiu ao OX. Já para as instâncias do PostVRP, que não possuíam resultados prévios para comparação, conseguimos os primeiros limitantes para o problema considerando instâncias com até 10.000 pontos de entrega. **Palavras-chave:** VRP, algoritmo genético, busca local.

Abstract

The Vehicle Routing Problem (VRP) is a generalization of the classic Traveling Salesman Problem (TSP) and it can be broadly defined as a class of problems about visiting "costumers" through "vehicles". Such problem appeared in 1959 and has a challenging structure to be optimized. The VRP models several real situations, like vehicles routing for dispatch calls, establishing routes for mailmen, establishing routes for garbage collection, establishing itineraries for buses, and many others. This work developed genetic algorithms (AGs) combined to local search and applied them to two problem variants, with the primary focus being to minimize the number of vehicles. The first variant models a real case based on the post offices of Artur Nogueira, SP, with the name PostVRP. It is a large scale benchmark, with up to 30,000 delivery points, which has unknown bounds. The second variant is called Capacitated Vehicle Routing Problem (CVRP) and the instances set used here already has known optimal values for comparison purposes. The experiments conducted had four genetic algorithms variations: employing Best Cost Route Crossover (BCR) or Order Based Crossover (OX) and running the AG with or without local search. The proposed algorithms reached the known optimal results in 46% of the CVRP instances, and were more than 10% worse in only 9%. The use of local search combined to the AG accelerated the solutions convergence and, between the two tested crossover operators, BCR surpassed OX. For the PostVRP instances, which had no previous results for comparison, we obtained the first bounds for the problem considering up to 10,000 delivery points. **Keywords:** VRP, genetic algorithm, local search.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviações e Siglas

AG: Algoritmo Genético

BCR: Best Cost Route Crossover

CVRP: Capacitated Vehicle Routing Problem

CWS: Clarke and Wright Savings Algorithm

DVS: Deterministic Vertex Swap

EAX: Edge Assembly Crossover

FJA: Fisher and Jaikumar Algorithm

MDVRP: Multi-Depot Vehicle Routing Problem.

OX: Order Based Crossover

PostVRP: Vehicle Routing Problem applied to Post Office.

RVS: Random Vertex Swap

SC: String Cross

SE: String Exchange

SM: String Mix

SR: String Relocation

TSP: Traveling Salesman Problem

VRP: Vehicle Routing Problem

VRPTW: Vehicle Routing Problem with Time Windows.

Sumário

Capítulo 1

Introdução

Mencionado pela primeira vez na literatura em 1959, o Problema de Roteamento de Veículos (do inglês *Vehicle Routing Problem*, VRP) foi proposto como uma generalização do Problema do Caixeiro Viajante (do inglês *Traveling Salesman Problem*, TSP) e inicialmente aplicado para otimizar as rotas de uma frota de caminhões de combustível (DANTZIG; RAMSER, 1959). Desde então, o VRP vem recebendo crescente interesse e importância para áreas de pesquisa como logística (CAO; YANG, 2017), além de ter se dividido em diversas variantes (FARAHANI; REZAPOUR; KARDAR, 2011) capazes de modelar situações reais que vão desde roteamento para caminhões de lixo (ASSAF; SALEH, 2017) até embarcações de abastecimento *offshore* (BORTHEN; LOENNECHEN; WANG; FAGERHOLT; VIDAL, 2018).

Duas variantes do VRP foram de interesse para este trabalho. A primeira delas foi o PostVRP, uma variante aplicada ao roteamento de entregas dos correios que foi proposta junto a uma ferramenta para geração de instâncias (MEIRA; MARTINS; MENZORI; ZENI, 2018). Tal ferramenta trata-se de um *benchmark* desenvolvido a partir de um caso real: a entrega de correspondências por carteiros a pé na cidade de Artur Nogueira, São Paulo. As instâncias deste *benchmark* são de grande escala (até 30.000 pontos), multi-objetivo e não possuíam limitantes conhecidos até o momento.

A segunda variante foi o CVRP (do inglês *Capacitated Vehicle Routing Problem*), a forma mais simples do problema, que conta com um limite de capacidade dos veículos. Um conjunto de instâncias com até 1.000 pontos de entrega, chamado conjunto X, foi proposto para oferecer uma diversidade maior de casos (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017).

As demais partes desta dissertação estão organizadas da seguinte forma: os objetivos deste trabalho foram descritos na Seção 1.1. O Capítulo 2 apresenta uma descrição formal do problema na Seção 2.1, seguida de algumas abordagens comuns para resolver o VRP nas Seções 2.2, 2.3 e 2.4. No Capítulo 3 temos um breve levantamento bibliográfico. No Capítulo 4 estão descritos os algoritmos implementados e as instâncias utilizadas nas Seções 4.1 e 4.2, respectivamente. O Capítulo 5 apresenta os experimentos realizados e os resultados obtidos nas Seções 5.1 até 5.4. Por fim, as conclusões se encontram no Capítulo 6.

1.1 Objetivos

Este trabalho teve como objetivo resolver o VRP através da combinação de duas abordagens: algoritmos genéticos (AG) e heurísticas de busca local. Mais especificamente, foram implementados um AG com dois operadores diferentes de *crossover*, e duas heurísticas de busca local.

Estes algoritmos foram usados para a obtenção dos primeiros limitantes para instâncias PostVRP. Já os resultados obtidos para instâncias CVRP foram comparados à literatura.

Para este trabalho o foco foi dado à minimização do número de veículos. Em caso de empate, foi utilizado o comprimento da menor rota como fator de desempate.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta uma descrição formal do problema e as definições das variantes tratadas neste trabalho, seguido por algumas abordagens comuns encontradas na literatura para o VRP.

2.1 Descrição Formal do Problema

Esta seção foi baseada no trabalho (MEIRA; MARTINS; MENZORI; ZENI, 2018). Considere um grafo ponderado $G(V, E)$ e uma função custo $w' : E \to \mathbb{Q}^+$. O grafo pode ser orientado ou não. Existe um vértice especial $\pi \in V$ chamado depósito. Existem variantes do VRP com múltiplos depósitos, as quais não serão tratadas nesta seção. O conjunto de clientes é dado por $C = V \setminus \{\pi\}$ e o número de clientes é dado por $n = |C|$. O conjunto de clientes será representado por $C = \{c_1, \ldots, c_n\}$. Existe um valor $k \in \mathbb{N}$ que representa o número de veículos. O valor de k pode ser uma constante, parte da entrada, ou pode ser uma variável definida no processo de otimização. Seja $w(u, v, G)$ o custo do menor caminho entre os vértices u e v no grafo G. Se o grafo ponderado G estiver claro no contexto, pode-se usar $w(u, v)$ para representar $w(u, v, G)$.

Em sua forma mais básica do problema, o VRP envolve usar os veículos k para visitar todos os clientes do conjunto C minimizando uma função objetivo(TOTH; VIGO, 2002a). No trabalho de (MEIRA; MARTINS; MENZORI; ZENI, 2018) uma sequência de elementos foi usada para representar uma solução deste problema, conforme descrito abaixo:

$$\mathcal{S}(C,k) = (c_1, \ldots, c_n, \pi, \ldots, \pi).$$

Esta sequência é montada da seguinte maneira. Primeiro, todos os clientes são inseridos em $\mathcal{S}$. Após isso, o vértice depósito é inserido $k-1$ vezes. Se o conjunto de clientes C e o número de veículos k forem conhecidos, poderemos usar $\mathcal{S}$ para representar $\mathcal{S}(C,k)$.

Cada permutação de $\mathcal{S}$ representa uma solução do VRP. Para exemplificar, considere o grafo ilustrado na Figura 2.1. Considere os vértices descritos por (a) na Figura 2.1. Considere o número de veículos igual a 3, ou seja $k = 3$. A sequência $\mathcal{S}(C,3)$ neste caso seria: $\mathcal{S}(C,3) = (c_1, \ldots, c_{13}, \pi, \pi)$.

Cada permutação de $\mathcal{S}(C,3)$ representa uma solução para o VRP. Por exemplo, a permutação $\mathcal{S}' = (c_3, c_5, c_4, c_1, c_2, \pi, c_6, c_{10}, c_{11}, c_{12}, \pi, c_7, c_8, c_9, c_{13})$ representa a solução descrita por (b) na Figura 2.1.

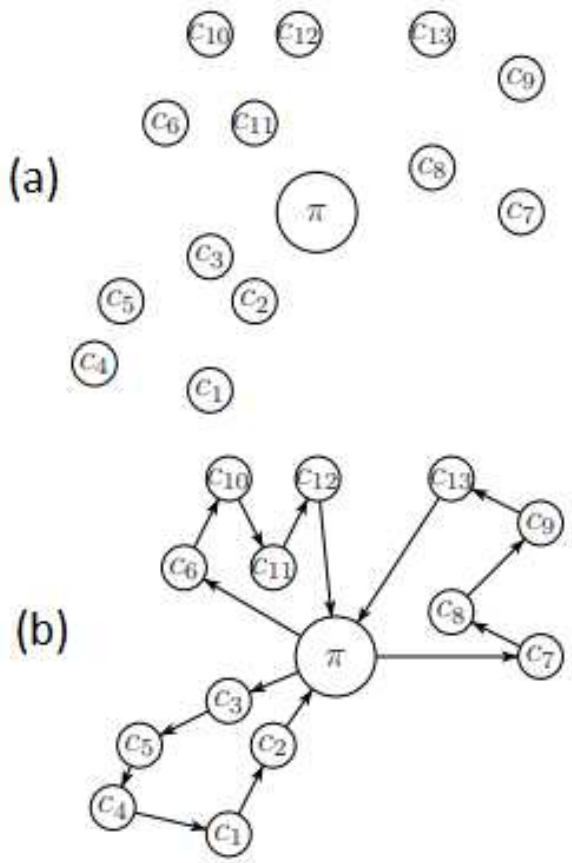


Figura 2.1: Exemplo de Instância do VRP (a), seguido de uma possível solução $\mathcal{S}' = (c_3, c_5, c_4, c_1, c_2, \pi, c_6, c_{10}, c_{11}, c_{12}, \pi, c_7, c_8, c_9, c_{13})$ (b). (Imagem adaptada de (MEIRA; MARTINS; MENZORI; ZENI, 2018))

Toda rota inicia e termina no depósito. A solução $\mathcal{S}'$ representa uma partição dos clientes em 3 rotas: $\mathcal{R}_1 = (c_3, c_5, c_4, c_1, c_2)$, $\mathcal{R}_2 = (c_6, c_{10}, c_{11}, c_{12})$ e $\mathcal{R}_3 = (c_7, c_8, c_9, c_{13})$. Mais formalmente, uma solução pode ser particionada em diversas rotas onde $k' \leq k$:

$$Particao(\mathcal{S}) = (\mathcal{R}_1, \ldots, \mathcal{R}_{k'}),$$

onde a partição da solução é feita no depósito π. No exemplo da Figura 2.1 temos que: $Particao(\mathcal{S}') = (\{c_3, c_5, c_4, c_1, c_2\}, \{c_6, c_{10}, c_{11}, c_{12}\}, \{c_7, c_8, c_9, c_{13}\})$. Por definição, rotas vazias não fazem parte de $Particao(\mathcal{S})$. Assim $Particao(1, 2, \pi, \pi, 3, 4)$ é $\{(1, 2), (3, 4)\}$ e não $\{(1, 2), (), (3, 4)\}$, portanto $k' \leq k$.

O comprimento de uma rota $\mathcal{R} = (r_1, \ldots, r_m)$ é dado por:

$$W(\mathcal{R}) = w(\pi, r_1) + w(r_m, \pi) + \sum_{i=1}^{m-1} w(r_i, r_{i+1}).$$

O comprimento de uma solução $\mathcal{S} = (s_1, \ldots, s_m)$ é calculado de pelo somatório do comprimento das rotas:

$$f_1\mathcal{S} = \sum_{\mathcal{R} \in Particao(\mathcal{S})} (W(\mathcal{R})).$$

O número de veículos usados em uma solução é igual ao número de rotas não vazias $|Particao(\mathcal{S})|$. Se o número de veículos é k e não são permitidas rotas vazias, temos a restrição $|Particao(\mathcal{S})| = k$. Se o número de veículos é no máximo k, ou se rotas vazias são permitidas, nós temos $|Particao(\mathcal{S})| \leq k$. Se o número de veículos k não é parte da entrada, o domínio pode ser representado pela permutação da sequência $\mathcal{S}(C, n)$. Neste caso, o número de veículos é definido durante o processo de otimização.

Dada uma solução viável, (MEIRA; MARTINS; MENZORI; ZENI, 2018) trabalham de maneira multiobjetivo, com três custos. O valor mais tradicional a ser minimizado é o comprimento de uma solução,

$$f_1(\mathcal{S}) = W(\mathcal{S}).$$

Outra função objetivo consiste em minimizar o número de veículos.

$$f_2(\mathcal{S}) = |Particao(\mathcal{S})|.$$

Finalmente, é necessário gerar rotas justas, ou seja, rotas cujos comprimentos sejam balanceados de modo que não penalizem um veículo em relação a outro. Uma maneira de representar a justiça é por meio do cálculo da variância entre o comprimento das rotas de uma dada solução:

$$f_3(\mathcal{S}) = \sqrt{\frac{\sum_{\mathcal{R} \in Particao(\mathcal{S})} \left(W(\mathcal{R}) - \overline{W(\mathcal{R})}\right)^2}{|Particao(\mathcal{S})| - 1}}.$$

(MEIRA; MARTINS; MENZORI; ZENI, 2018) consideram uma variante do problema onde o tamanho da rota é limitada. Este problema pode modelar veículos que precisam reabastecer no depósito, como um helicóptero por exemplo. Também pode modelar situações trabalhistas, onde o condutor de um veículo tem um limite de disponibilidade de tempo. Considere que existe um limite $\mathcal{R}_{max}$ para o comprimento da rota. Desta forma, toda rota $\mathcal{R}$ precisa respeitar a seguinte restrição para ser considerada viável:

$$W(\mathcal{R}) \leq \mathcal{R}_{max}.$$

Abaixo temos a definição do problema de interesse, que será chamado de PostVRP, ou seja, VRP no contexto dos correios *(Post Office)*, assim como a definição de um segundo problema também abordado neste trabalho, o CVRP.

Definição 1 (PostVRP) *Dado um grafo ponderado $G(V, E)$, uma constante k representando o número máximo de veículos, um vértice especial $\pi \in V$ e o comprimento máximo de uma rota $\mathcal{R}_{max}$. Seja $C \leftarrow V \setminus \{\pi\}$. Considere a sequência $\mathcal{S}(C, k)$. Seja P o conjunto de todas as permutações de $\mathcal{S}(C, k)$. Encontre as permutações $\mathcal{S}^* \subset P$ viáveis em relação a $\mathcal{R}_{max}$ e que minimizem as funções objetivos $f_1(\mathcal{S})$, $f_2(\mathcal{S})$ e $f_3(\mathcal{S})$.*

Além do PostVRP, trabalhamos com o CVRP (TOTH; VIGO, 2002a). Suponha que cada ponto de entrega $r \in C$ possui uma demanda $d(r) \in \mathbb{R}^+$. Considere uma rota

$\mathcal{R} = (r_1, \ldots, r_m)$. A demanda de uma rota é igual a

$$D(\mathcal{R}) = \sum_{r \in \mathcal{R}} d(r_i)$$

Definição 2 (CVRP) *Dado um grafo ponderado $G(V, E)$, um vértice especial $\pi \in V$, uma demanda $d(r) \in \mathbb{N}$ para todo $r \in C$ e a capacidade máxima de um veículo $D_{max} \in \mathbb{N}$. Seja $C \leftarrow V \setminus \{\pi\}$ com $n = |C|$. Considere a sequência $\mathcal{S}(C, n)$. Seja P o conjunto de todas as permutações de $\mathcal{S}(C, n)$. Encontre as permutações $\mathcal{S}^* \subset P$ viáveis em relação a $D(\mathcal{R}) \leq D_{max}$ para toda rota $\mathcal{R}$ e que minimizem as funções objetivos $f_1(\mathcal{S})$, $f_2(\mathcal{S})$.*

Nesta pesquisa trabalhamos no caso restrito onde foi minimizado f_2. Em caso de empate em relação a f_2, utilizamos o comprimento da menor rota como fator de desempate. Note que, no PostVRP, minimizar f_2 está correlacionado com minimizar f_1 uma vez que, para reduzir o número de rotas é necessário reduzir o comprimento das rotas.

Nas próximas subseções iremos descrever as técnicas de otimização comumente utilizadas para resolver VRP.

2.2 Métodos Diretos

Diferente de meta-heurísticas, as quais são suficientemente genéricas para serem aplicadas a uma grande variedade de problemas, os métodos diretos são projetados de forma mais focada para serem aplicados a um problema específico.

Também conhecidos como heurísticas clássicas, os métodos diretos para o VRP foram desenvolvidos principalmente entre 1960 a 1990. Capazes de

produzir boas soluções em tempos computacionais modestos, eles executam uma exploração mais limitada do espaço de busca e podem ser facilmente adaptados à restrições encontradas no mundo real.

Heurísticas clássicas podem ser divididas em três categorias, as quais serão abordadas nas subseções 2.2.1, 2.2.2 e 2.2.3. São elas: métodos construtivos, que vão construindo soluções viáveis aos poucos; métodos em duas fases, que dividem o problema em agrupamento de vértices e construção de rotas; e métodos incrementais, que tentam melhorar soluções iniciais através de trocas de vértices ou arestas.

2.2.1 Construtivos

O algoritmo construtivo mais conhecido para o VRP é provavelmente o *Clark and Wright Savings Algorithm* (CWS) (CLARKE; WRIGHT, 1964). O algoritmo se baseia na união de duas rotas de modo que a rota resultante gere uma redução de custo, chamada ganho ou *saving*. Particularmente indicado para problemas sem um número fixo de veículos, o CWS pode ser implementado em duas versões: *(i)* sequencial, onde é construída uma rota de cada vez e *(ii)* paralela, onde várias rotas são criadas ao mesmo tempo.

O Algoritmo 1 é o CWS paralelo. Tal algoritmo inicia com n rotas, cada uma partindo do depósito, visitando um único ponto de entrega e retornando ao depósito. Desta forma, o custo da solução começa bastante alto.

Para cada par de vértices há um valor associado que representa o ganho ao se juntar duas rotas. Para calcular este ganho, considere dois pontos de entrega d_1 e d_2 e considere duas rotas $r_1 = (\pi, d_1, \pi)$ e $r_2 = (\pi, d_2, \pi)$. Ao se

unir tais rotas teremos $r_3 = (\pi, d_1, d_2, \pi)$. Sendo assim, o ganho é dado por:

$$s(d_1, d_2) = w(\pi, d_1) + w(\pi, d_2) - w(d_1, d_2). \tag{2.1}$$

Note que $s(d_1, d_2)$ é igual ao $custo(r_1)+custo(r_2)-custo(r_3)$. Uma vantagem do CWS é que o valor $s(d_1, d_2) = w(\pi, d_1) + w(\pi, d_2) - w(d_1, d_2)$ é fixo e pode ser calculado uma única vez no início do algoritmo.

Um detalhe importante, é que o conceito de *saving* envolvendo rotas maiores, permite a união apenas nos extremos. Seja *prefixo* e *sufixo* dois conjuntos quaisquer de entregas, considere duas rotas $r_1 = (\pi, prefixo, d_1, \pi)$ e $r_2 = (\pi, d_2, sufixo, \pi)$. Ao se unir tais rotas teremos $r_3 = (\pi, prefixo, d_1, d_2, sufixo, \pi)$. O ganho será o mesmo mostrado na Equação 2.1.

Em suma, as rotas iniciam com tamanho um e vão sendo incrementadas. A cada iteração, um vértices ou uma rota é concatenada a outra. Note que a união só é feita se respeitar à restrições, como as de capacidade por exemplo.

Na versão sequencial, existe uma única rota que é incrementada em um vértice a cada iteração. A escolha é sempre do vértice que ofereça o maior ganho. Ao atingir a capacidade do veículo, a rota atual é fechada e uma nova rota é iniciada.

Já na versão paralela, várias rotas são criadas ao mesmo tempo. Sempre são concatenadas as duas rotas tais que seus extremos s_1 e s_2 possuam o máximo *saving* $s[s_1][s_2]$, considerando todos os pares de extremos possíveis.

Esta heurística tem uma boa qualidade no início das rotas, colocando sempre um novo elemento promissor. Entretanto as rotas tendem a crescer deixando pontos para trás. Tais pontos acabam depois sendo cobertos por meio de rotas ruins, de alto custo. A Figura 2.2 ilustra as etapas do algoritmo.

1. Calcular os *savings* para cada par de vértices;

2. ordenar os *savings* em ordem decrescente;

3. para cada *saving*, unir os vértices envolvidos e suas respectivas rotas caso possam ser unidos;

4. se algum vértice não pôde ser unido durante o passo 3, tal vértice se torna uma rota por si só.

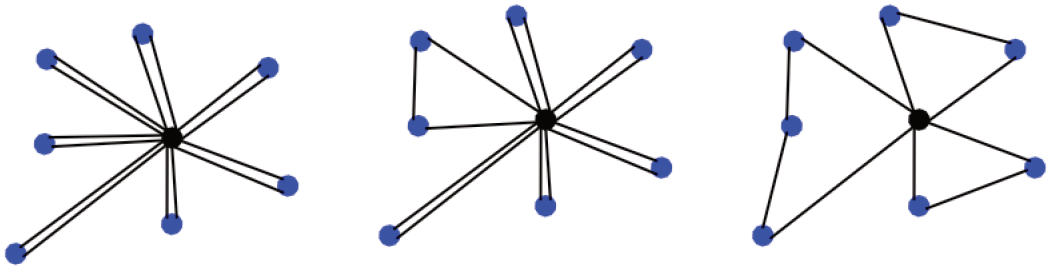
Algoritmo 1: Algoritmo CWS paralelo para o CVRP. Adaptado de (TOTH; VIGO, 2002b)

Entrada: Um grafo $G(V, E)$ com pesos $w : E \to \mathbb{R}^+$, um inteiro $k \in \mathbb{N}$, um depósito $\pi \in V$ e o comprimento máximo de uma rota $\mathcal{R}_{max} \in \mathbb{R}^+$. Seja $C = V \setminus \{\pi\}$.

Saída: Uma permutação $\mathcal{S}(C, k) \in P$

```
Calcule s[u][v] para todo u ≠ v em V.
Seja S = {s[u][v] | (u, v) ∈ V × V, u ≠ v}
Seja S' uma ordenação decrescente de S.
Seja Sol um conjunto de rotas, inicialmente Sol = C, ou seja cada vértice é uma
  rota com um único elemento
for i = 1, i ≤ |S'|, i++, do
    Considere o saving s'_i ∈ S' e seus vértices u e v.
    if u for o extremo de uma rota R' ∈ Sol e v for o extremo de outra rota
      R'' ∈ Sol then
        if custo R' ∪ R'' for viável then
            Sol ← Sol \ {R', R''}
            Sol ← Sol ∪ {R'R''}
        end
    end
end
return Sol;
```

O livro (TOTH; VIGO, 2001) ainda cita mais duas heurísticas construtivas, são elas o *Matching-Based Savings Algorithm* (ALTINKEMER; GAVISH, 1991) e a *Sequential Insertion Heuristic* (MOLE; JAMESON, 1976).



(a) Cada vértice começa visitado por uma rota.

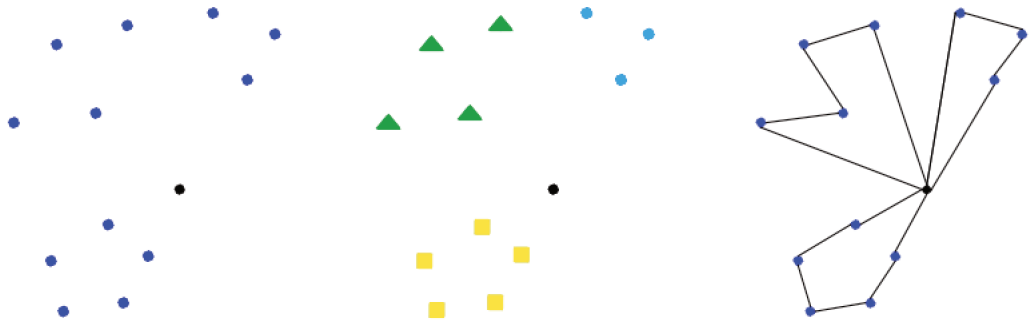
(b) Duas rotas são unidas pela concatenação dos vértices com maior *saving*.

(c) Solução resultante das uniões respeitando os limites de cada veículo.

Figura 2.2: Ação do CWS no grafo. Fonte: própria.

2.2.2 Duas Fases

As heurísticas em duas fases são divididas em duas abordagens, *cluster-first, route-second* ou *route-first, cluster-second*. Usando *cluster-first, route-second*, um conjunto de vértices é particionado em subconjuntos que então são roteados separadamente. Já usando *route-first, cluster-second*, uma grande rota é traçada para o conjunto completo de vértices e depois uma técnica de agrupamento é aplicada para a quebra do resultado em várias rotas menores. As figuras 2.3 e 2.4 ilustram estas duas abordagens.



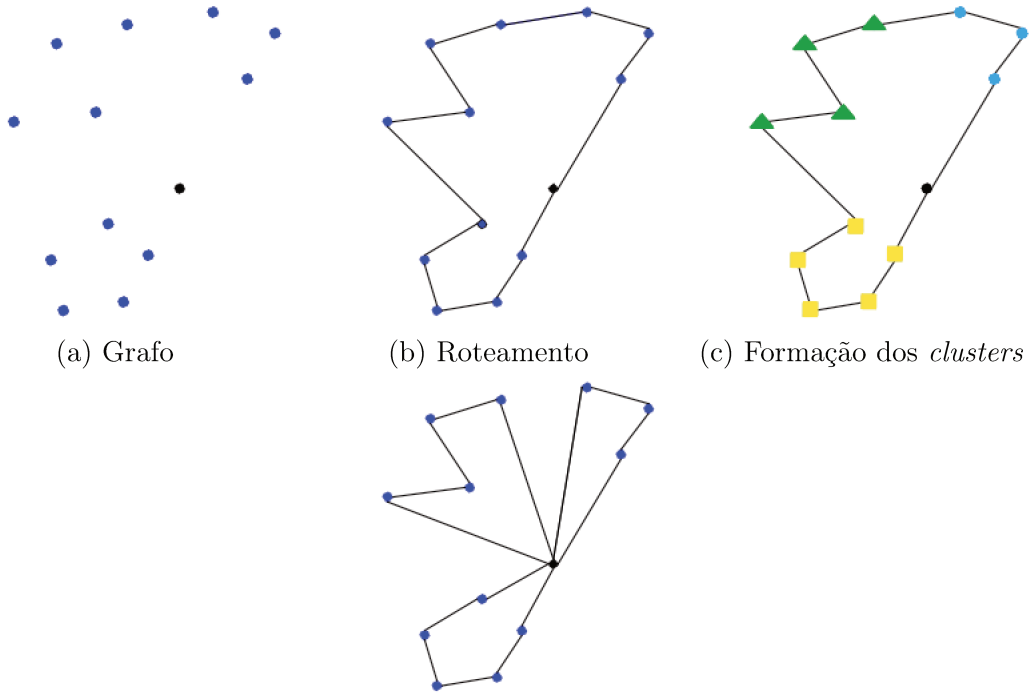
(a) Grafo

(b) Formação dos *clusters*

(c) Roteamento

Figura 2.3: VRP em duas fases: *cluster-first, route-second*. Fonte: própria.

Um dos algoritmos mais simples do tipo *Cluster-first* é o *Sweep Algorithm* (GILLETT; MILLER, 1974). Em tal algoritmo, cada vértice a ser visitado possui coordenadas polares. Estes vértices são varridos por um raio



(a) Grafo (b) Roteamento (c) Formação dos *clusters*

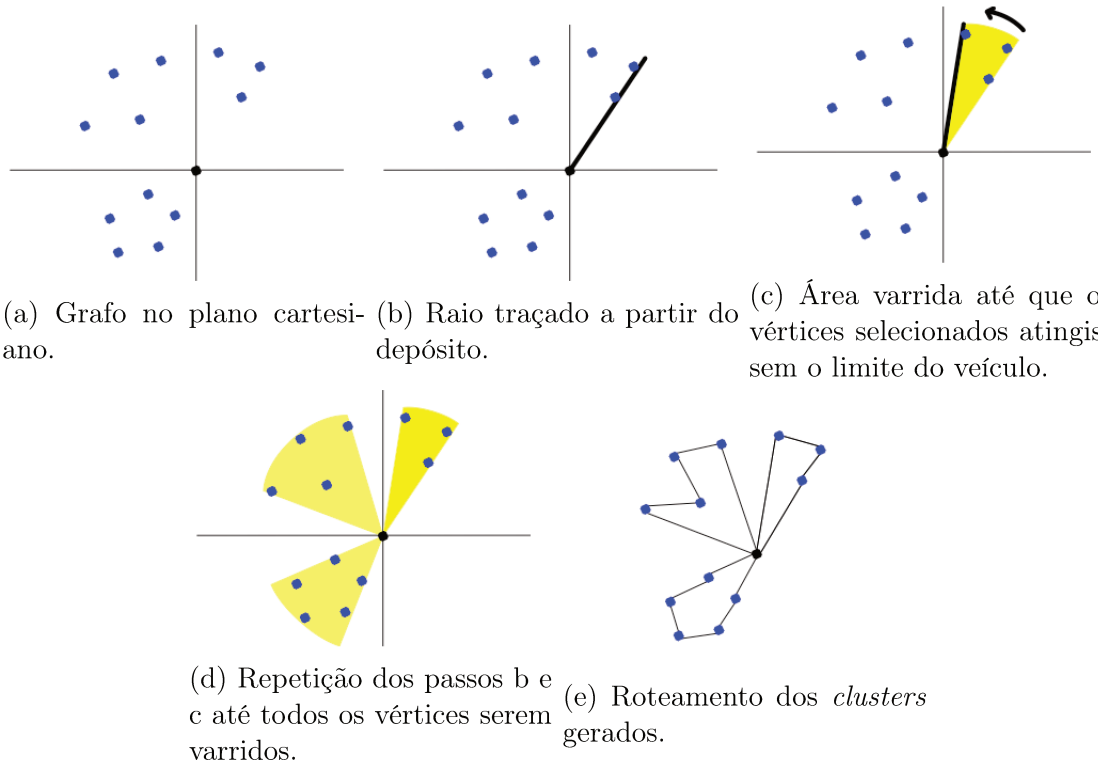
(d) Quebra em rotas viáveis.

Figura 2.4: VRP em duas fases: *route-first, cluster-second*. Fonte: própria.

traçado a partir do depósito. Esta varredura vai então atribuindo os vértices a um veículo em ordem crescente de ângulo até que o limite de capacidade do veículo seja atingido. O processo se repete até que todos os vértices pertençam a um veículo. Finalmente, cada rota gerada é otimizada individualmente como um TSP. A Figura 2.5 ilustra o procedimento aqui descrito.

Outros algoritmos *Cluster-first* citados em (TOTH; VIGO, 2001) são o *Fisher and Jaikumar Algorithm* (FJA) (FISHER; JAIKUMAR, 1981) e o *Petal Algorithm* (FOSTER; RYAN, 1976).

Em relação à abordagem *Route-first*, ela foi proposta pela primeira vez por Beasley (BEASLEY, 1983). Como a segunda etapa da abordagem pode ser vista como um problema de caminho mínimo em um grafo acíclico, é possível resolvê-la com o algoritmo de Dijkstra (DIJKSTRA, 1959), por exemplo. Considerando $custo(O, i) + custo(O, j) + l(i, j)$ usado para calcular o



(a) Grafo no plano cartesiano.

(b) Raio traçado a partir do depósito.

(c) Área varrida até que os vértices selecionados atingissem o limite do veículo.

(d) Repetição dos passos b e c até todos os vértices serem varridos.

(e) Roteamento dos *clusters* gerados.

Figura 2.5: Execução do *Sweep Algorithm*. Fonte: própria.

$custo(i, j)$ no problema de caminho mínimo, o valor de $l(i, j)$ corresponde ao custo de viajar de i até j no percurso gerado na etapa de roteamento.

Segundo Toth (TOTH; VIGO, 2001), não há estudos demonstrando métodos *Route-first* que sejam competitivos à outras técnicas.

2.2.3 Incrementais

Algoritmos incrementais são técnicas de busca local criadas especificamente para o VRP e se baseiam na melhora de soluções já existentes através de pequenas mudanças. Neste trabalho os termos "algoritmo incremental" e "busca local" são usados como sinônimos.

Caso cada rota do VRP seja trabalhada individualmente, qualquer técnica para TSP pode ser empregada. Caso contrário, também existem alguns métodos que trabalham com múltiplas rotas.

Entre as abordagens para rotas individuais, podemos citar três:

- Movimento de vértice: um ou mais vértices são movidos para outra posição;
- Troca de vértices: um par de vértices trocam de posição;
- Troca de arestas: duas ou mais arestas trocam de posição.

A troca de arestas é também conhecida como λ-opt(LIN, 1965). O algoritmo remove λ arestas da solução e tenta reconectar os segmentos resultantes de modo a reduzir o custo da rota. Tal processo se repete até que um ótimo local seja atingido. O algoritmo 2 descreve uma implementação do 2-opt.

Algoritmo 2: Implementação do 2-opt.

```
while houve melhora do
    for cada par de arestas não consecutivas do
        encontrar o par cuja troca de posição melhor reduz o custo da solução;
    end
    arestas são trocados de posição;
end
```

Executando em tempo $O(n^{\lambda})$, suas variantes mais comuns são o 2-opt e 3-opt, troca de 2 e 3 arestas, respectivamente. Johnson e McGeoch (JOHNSON; MCGEOCH, 1997) analisaram o desempenho de algoritmos incrementais para o TSP e concluíram que o melhor era uma implementação do λ-opt com λ dinâmico (LIN; KERNIGHAN, 1973).

Entre as abordagens que trabalham com múltiplas rotas, Thompson e Psaraftis (THOMPSON; PSARAFTIS, 1993) descreveram uma técnica geral conhecida como *b-cyclic, k-transfer exchanges*, na qual k clientes de uma rota são movidos para a rota seguinte em um ciclo de permutações com b rotas. Dentro desta premissa, Breedam (BREEDAM, 1994) classifica 4 tipos de operações: *String cross* (SC), na qual duas *strings* de vértices são trocadas cruzando arestas entre duas rotas; *String exchange* (SE), na qual duas *strings*

com um número máximo de k vértices são trocadas entre duas rotas; *String relocation* (SR), na qual uma *string* com um número máximo de k vértices é transferida a outra rota; e *String mix* (SM), na qual é feita a melhor operação entre SE e SR.

2.3 Algoritmos Genéticos e Bio-inspirados

Algoritmos bio-inspirados são soluções computacionais desenvolvidas a partir de comportamentos observados na natureza (HOROWITZ; SAHNI, 1978). A seleção natural, a busca de comida por formigas e as sinapses cerebrais são alguns exemplos de "mecanismos" naturais relativamente bem conhecidos pela biologia que funcionam bem para seus respectivos propósitos e portanto servem de inspiração para heurísticas.

A implementação de tais algoritmos é obviamente uma simplificação da realidade que exige uma boa adaptação ao caso em questão e não garante resultados ótimos.

Algoritmos genéticos/evolutivos, por exemplo, possuem as seguintes etapas (BACK, 1996):

1. Gerar uma população inicial de soluções;
2. Filtrar as soluções através de "pressões ambientais";
3. Cruzar as soluções sobreviventes;
4. Introduzir aleatoriamente um "fator de mutação";
5. Obter a nova geração de soluções e
6. Repetir o processo a partir da etapa 2.

No caso de algoritmos genéticos, cada permutação da sequência $\mathcal{S}(C, k)$ pode representar um indivíduo da população. Existem diversas possibilidades de cruzamentos entre dois indivíduos. A regra consiste em gerar um vetor $\mathcal{S}^3$ válido a partir de dois vetores $\mathcal{S}^1$ e $\mathcal{S}^2$.

No Capítulo I do livro (PEREIRA; TAVARES, 2008) foi descrito uma metodologia para cruzamento de soluções do TSP adaptada com sucesso para o VRP. Tal técnica chama-se *EAX crossover* (*Edge Assembly Crossover*).

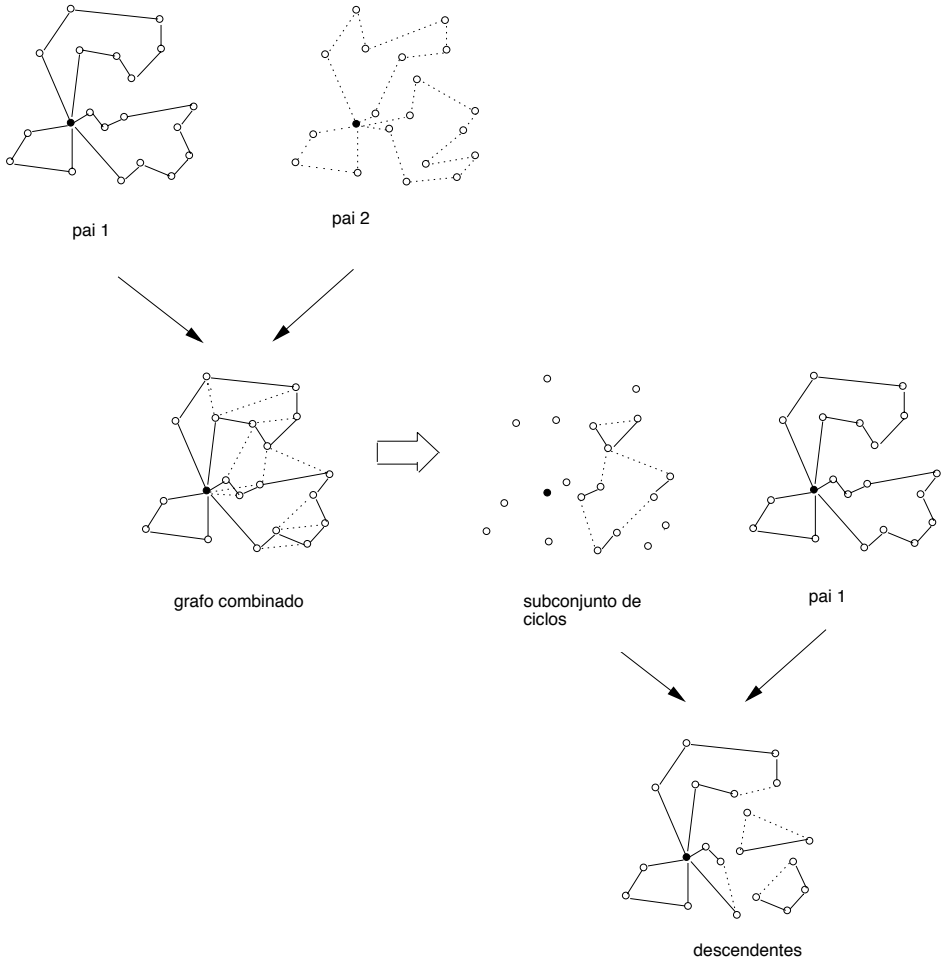


Figura 2.6: *EAX crossover*. Imagem adaptada de (PEREIRA; TAVARES, 2008).

Como ilustrado na Figura 2.6, tal algoritmo começa gerando um grafo que tenha todas as arestas obtidas pela união de duas soluções pai. Subconjuntos de ciclos resultantes desta combinação são então particionados e um deles, assim como um dos pais, são selecionados para a etapa seguinte. O pai

escolhido tem suas arestas presentes no subconjunto removidas enquanto as demais arestas do subconjunto lhe são adicionadas, criando uma solução intermediária. Caso o grafo descendente esteja desconexo, uma abordagem gulosa é utilizada para torná-lo conexo. Finalmente, violações de restrições são eliminadas e técnicas de troca de vértices ou λ-opt são aplicadas até que a solução seja viável.

2.4 Outros

Existem demais técnicas de otimização como outras heurísticas de busca local (CESCHIA; SCHAERF; STÜTZLE, 2013; MLADENOVIĆ; HANSEN, 1997), *simulated annealing* e busca tabu (OSMAN, 1993). Tais técnicas procuram varrer o domínio de soluções em direções de melhora da função objetivo. Cada permutação de $\mathcal{S}(C, k)$ é uma solução possível e soluções próximas são consideras vizinhas. Tais técnicas varrem parte do domínio, saltando de solução em solução, buscando melhorar a função objetivo.

A técnica de *branch-and-bound* também pode ser usada para o VRP (PECIN, D. G., 2014). Tal técnica pertence a classe de algoritmos exatos. Ela varre, de maneira organizada, o domínio de solução, onde grandes conjuntos são removidos no processo de poda. Com complexidade exponencial para encontrar a solução ótima, o *branch-and-bound* possui um *gap* entre a melhor solução encontrada e um limitante inferior para o ótimo. O *gap* pode ser útil mesmo que o *branch-and-bound* não execute até o fim. Algoritmos genéticos e a maioria das heurísticas descritas neste projeto não fornecem nenhuma informação de quão longe a solução se encontra do ótimo.

Capítulo 3

Levantamento bibliográfico

O TSP é um dos mais estudados da otimização combinatória (APPLEGATE; COOK; ROHE, 2003; APPLEGATE; BIXBY; CHVATAL; COOK, 2006; COOK, 2011; LAWLER; LENSTRA; KAN; SHMOYS, 1985). Uma generalização largamente estudada do TSP é o Problema de Roteamento de Veículos, também conhecido como *Vehicle Routing Problem (VRP)* (TOTH; VIGO, 2001). Neste problema, deseja-se encontrar k ciclos que cubram todos os pontos de entrega (clientes) e que possuam um vértice especial em comum (depósito). O VRP possui requisitos adicionais de acordo com a situação.

Uma das primeiras aparições do VRP foi em 1959 (DANTZIG; RAMSER, 1959) sob o nome de *Truck Dispaching Problem*, uma generalização do Problema do Caixeiro Viajante. O nome VRP aparece em 1976 no trabalho de Christofides (CHRISTOFIDES, 1976). Christofides define VRP como um nome genérico dado a uma classe de problemas que envolvem visitar "clientes" com "veículos".

Situações distintas levam a diversas variantes do problema. Assumindo-se capacidade no veículo, temos o *Capacitated-VRP (CVRP)* (FUKASAWA; LONGO; LYSGAARD; ARAGÃO; REIS; UCHOA; WERNECK, 2006). Definindo-se janelas temporais de entrega, temos o *VRP with Time Windows*

(VRPTW) (KALLEHAUGE; LARSEN; MADSEN; SOLOMON, 2005). Havendo mais de um depósito, temos o *Multi-Depot VRP (MDVRP)* (RENAUD; LAPORTE; BOCTOR, 1996). Em casos de problemas que envolvem planejamento de rotas ao longo de vários dias, temos o *Periodic* VRP (PVRP) (FRANCIS; SMILOWITZ; TZUR, 2008). Outras variantes são facilmente encontradas na literatura.

O VRP modela diversas situações reais, como o roteamento de veículos para atender chamadas ou ocorrências, definição de rotas para carteiros, definição de rotas para coleta de lixo, definição de itinerários de ônibus fretados e vans, entre muitas outras. Ele é um problema da classe NP-Difícil (CRESCENZI; KANN; HALLDÓRSSON; KARPINSKI; WOEGINGER, 1997), o que implica não existirem algoritmos eficientes para resolvê-lo na exatidão a menos que $P = NP$.

Solomon (SOLOMON, 1987), em 1987, criou um benchmark para VRP com janelas temporais de entrega. São 6 conjuntos de problemas num total de 56 instâncias, nas quais o número de clientes é sempre 100. Os veículos possuem capacidade de entregas e os clientes possuem demanda ou peso. O número de veículos não é fixo, mas deriva do fato de haver um limite de capacidade. Neste sentido, o problema pode ser considerado multi-objetivo. Deseja-se minimizar o percurso e o número de veículos.

Em 2014, Uchoa et. al. criou uma biblioteca chamada CVRPLib (UCHOA; PECIN, D.; PESSOA; POGGI; SUBRAMANIAN; VIDAL, 2015). Nesta biblioteca, eles consolidaram as instâncias para o CVRP dos trabalhos (AUGERAT; BELENGUER; BENAVENT; CORBERÁN; NADDEF; RINALDI, 1995; CHRISTOFIDES; EILON, 1969; CHRISTOFIDES, 1979; FISHER, 1994; GOLDEN, B. L.; WASIL, E. A.; KELLY; CHAO, 1998;

LI; GOLDEN, B.; WASIL, E., 2005). Além disso, Uchoa et. al (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017) criaram novas instâncias com número de clientes variando de 100 a 1000.

Kim et. al (KIM; ONG; HENG; TAN; ZHANG, 2015) fizeram um levantamento do estado da arte sobre City VRP. Nesta variante existem restrições como, por exemplo, tráfego, poluição e vagas para estacionar. O trabalho categoriza mais de cem artigos sobre o VRP, levando em conta a logística da cadeia de produção em centros urbanos.

A tese de doutorado de Diego Pecin (PECIN, D. G., 2014), defendida no departamento de informática da PUC-RJ em 2014, propõem algoritmos exatos para o CVRP. Nela, os ótimos das instâncias `M-n200-k16` e `M-n200-k17` foram encontrados pela primeira vez, por meio de técnicas combinadas de geração de cortes e geração de colunas (*Branch-Cut-and-Price*). Tais instâncias foram propostas por Christofides, Mingozzi e Toth em 1976 (CHRISTOFIDES, 1976).

O VRP também pode ser combinado com problemas de empacotamento. Em 2016, A tese de doutorado de Pedro Hokama (HOKAMA et al., 2016), defendida no IC-Unicamp, propôs algoritmos para o VRP combinado com empacotamento de paralelepípedos em *containers*.

Diversos outros trabalhos tratam de roteamento combinado com empacotamento (GENDREAU; IORI; LAPORTE; MARTELLO, 2008; ZACHARIADIS; TARANTILIS; KIRANOUDIS, 2009; MÄNNEL; BORTFELDT, 2016). Gendreau (GENDREAU; IORI; LAPORTE; MARTELLO, 2008) usa busca tabu para resolver o VRP combinado com empacotamento 2D. Zachariadis (ZACHARIADIS; TARANTILIS; KIRANOUDIS, 2009) utiliza metaheurísticas como busca tabu associada a busca local para problemas de

roteamento combinado com empacotamento 2D. Dirk (MÄNNEL; BORTFELDT, 2016) cria algoritmos híbridos para solucionar roteamento combinado com empacotamento 3D.

Em 2008 foi lançado o livro *Bio-inspired algorithms for the vehicle routing problem* (PEREIRA; TAVARES, 2008). Tal livro apresenta uma visão geral dos algoritmos bio-inspirados aplicados ao VRP.

Existem vários trabalhos na literatura sobre o uso de algoritmos genéticos (AG) para resolver VRPs. Em 2017 um AG foi aplicado a um VRP baseado no serviço de coleta de lixo de uma cidade na palestina (ASSAF; SALEH, 2017). Em tal problema as rotas precisam sempre incluir uma parada obrigatória (o aterro) antes de retornarem ao depósito e, ao invés de demandas exatas, os pontos de coleta podem estar mais ou menos cheios dependendo da produção de lixo do dia.

Em 2012, (VIDAL; CRAINIC; GENDREAU; LAHRICHI; REI, 2012) propuseram um AG híbrido para resolver o VRP com mais de um depósito e o VRP periódico. Tal abordagem combina a exploração ampla do domínio, o uso de busca local e a manutenção da diversidade. Isso é feito de duas formas: (i) incluindo uma otimização por busca local logo após a etapa de *crossover*; (ii) considerando a contribuição dos indivíduos para a diversidade da população, o que é feito durante a seleção de soluções para a etapa de cruzamento e durante a seleção de soluções para sobreviver ao fim de cada geração.

Em 2018, foi feito um estudo para resolver um problema de planejamento de embarcações de abastecimento baseado em um caso real (BORTHEN; LOENNECHEN; WANG; FAGERHOLT; VIDAL, 2018). Por considerarem o problema similar a um VRP periódico, os autores decidiram usar uma

implementação do AG genético híbrido do trabalho (VIDAL; CRAINIC; GENDREAU; LAHRICHI; REI, 2012) com adaptações para considerar características particulares a realidade de transportes marítimos, como a longa duração das viagens.

Em 2015, um trabalho foi publicado compilando vários estudos sobre o uso de algoritmos genéticos para a resolução do *multi-depot* VRP (KARAKATIČ; PODGORELEC, 2015). Tal trabalho inclui as diferentes representações de solução, as formas de se criar a população, os operadores de *crossover* e mutação mais usados, os tipos de *fitness* mais comuns, as estratégias de seleção e os critérios de parada mais presentes na literatura. Além disso, os autores também apresentam um estudo comparativo entre algumas dessas várias alternativas. Os autores testaram cinco táticas de seleção. Destas cinco, o ranqueamento linear e torneio binário foram as de maior sucesso. Para operadores de *crossover*, entre 5 implementações, o destaque foi para o *Order Based* e o *Partially Matched*. Entre 7 operadores de mutação, os menos intrusivos, como a troca de dois vértices, renderam os melhores resultados.

Capítulo 4

Metodologia

Neste capítulo será descrita a abordagem empregada para a implementação dos algoritmos testados neste trabalho. A Seção 4.1 detalha como foi implementado cada componente do algoritmo genético, assim como os algoritmos de busca local que complementam o AG. Também neste capítulo, a Seção 4.2 descreve os conjuntos de instâncias utilizados.

Considere uma instância I do VRP. Seja o número de clientes n. O conjunto de clientes pode ser representado por $Del = (1, \ldots, n)$.

4.1 Algoritmo Genético

Para este trabalho foi implementado um algoritmo genético e dois algoritmos incrementais de troca de arestas. O AG possui as etapas clássicas da metaheurísta: criação da população inicial, seleção dos indivíduos que serão cruzados, *crossover*, mutação e seleção da nova população. Note que "indivíduo", "cromossomo" e "solução" são termos usados como sinônimos nesta seção.

Após execução de testes preliminares, foram definidos os parâmetros para o AG como mostra a tabela 4.1. Tamanho da população se refere à quan-

TAMANHO_POPULACAO	50
NUMERO_GERACOES	100
TAXA_CROSSOVER	0.95
TAXA_MUTACAO	0.10
TAMANHO_SELECAO_CANDIDATOS	50

Tabela 4.1: Parâmetros usados no AG.

tidade de soluções presentes no início e fim de cada geração. O número de gerações corresponde ao valor máximo de gerações que o AG pode atingir. As taxas de *crossover* e mutação são a probabilidade de que um indivíduo passará por estas etapas do AG. Tamanho da seleção de candidatos é o número de soluções consideradas para a etapa de *crossover*.

As soluções são sequências de inteiros, sendo um inteiro para cada ponto de entrega. Cada solução Sol é uma permutação do conjunto Del, ou seja $Sol \in Per(Del)$. Na representação escolhida, não há delimitação de rotas e o depósito não está incluído. As rotas são delimitadas a posteriori por um algoritmo guloso.

Demarcação de rotas: Como ilustrado na Figura 4.1, a criação de um cromossomo C é feita da seguinte maneira. Os elementos de Sol são inseridos sequencialmente no final de uma rota até que o valor limite para cada rota estabelecido pela instância não permita novas inserções. Quando isto ocorre, uma nova rota é criada e o processo se repete até todos os elementos serem inseridos. Feito isso, a solução passa a ser representada por sequências de inteiros intercaladas por zeros, que marcam o início/fim de cada rota. Note que o início e fim de cada rota ocorre no depósito. Note também que a demarcação de rotas é refeita sempre que o cromossomo é modificado.

A população inicial é criada através de permutações aleatórias de Del e a seleção dos candidatos ao *crossover* é feita por torneio. O torneio é feito

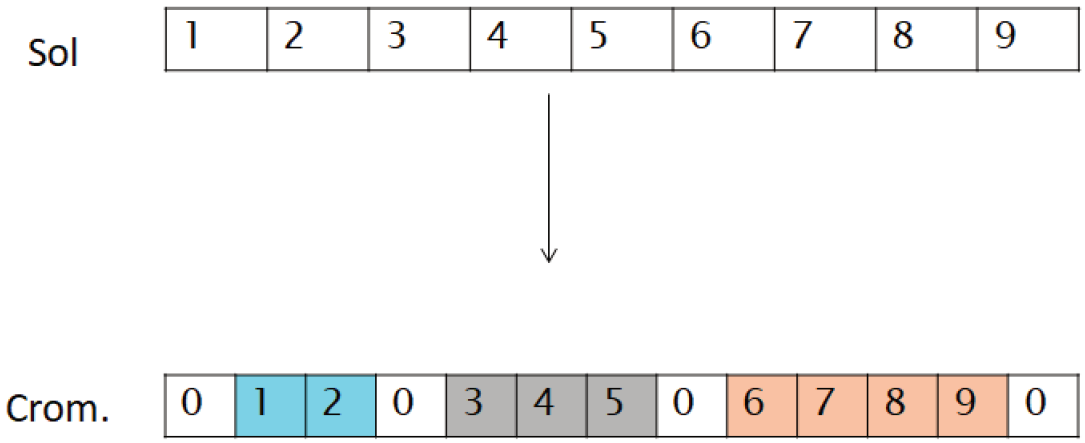


Figura 4.1: Processo de demarcação das rotas. Fonte: própria.

da seguinte maneira. São selecionados dois cromossomos aleatoriamente e aquele com melhor *fitness* entra na população de candidatos ao *crossover*. Isso se repete até que a população de candidatos atinja o valor pré-definido. É permitido repetição no grupo de candidatos.

Foram implementados dois operadores de *crossover*: *Best Cost Route Crossover* (BCR) (OMBUKI; ROSS; HANSHAR, 2006) e *Order Based Crossover* (OX) (OLIVER; SMITH; HOLLAND, 1987), que serão descritos a seguir.

O BCR crossover foi desenvolvido originalmente para problemas multi-objetivo de VRP com janelas temporais e foi aplicado neste trabalho sem nenhuma adaptação. Para cada par de soluções pai, os seguintes passos são tomados.

1. Uma rota é selecionada aleatoriamente de cada pai (Figura 4.2 a);
2. Os vértices pertencentes à rota selecionada de um pai são removidos do outro pai e vice-versa (Figura 4.2 b);
3. Os vértices removidos de cada pai são reinseridos, um por vez, de maneira independente, na melhor posição possível considerando todas as

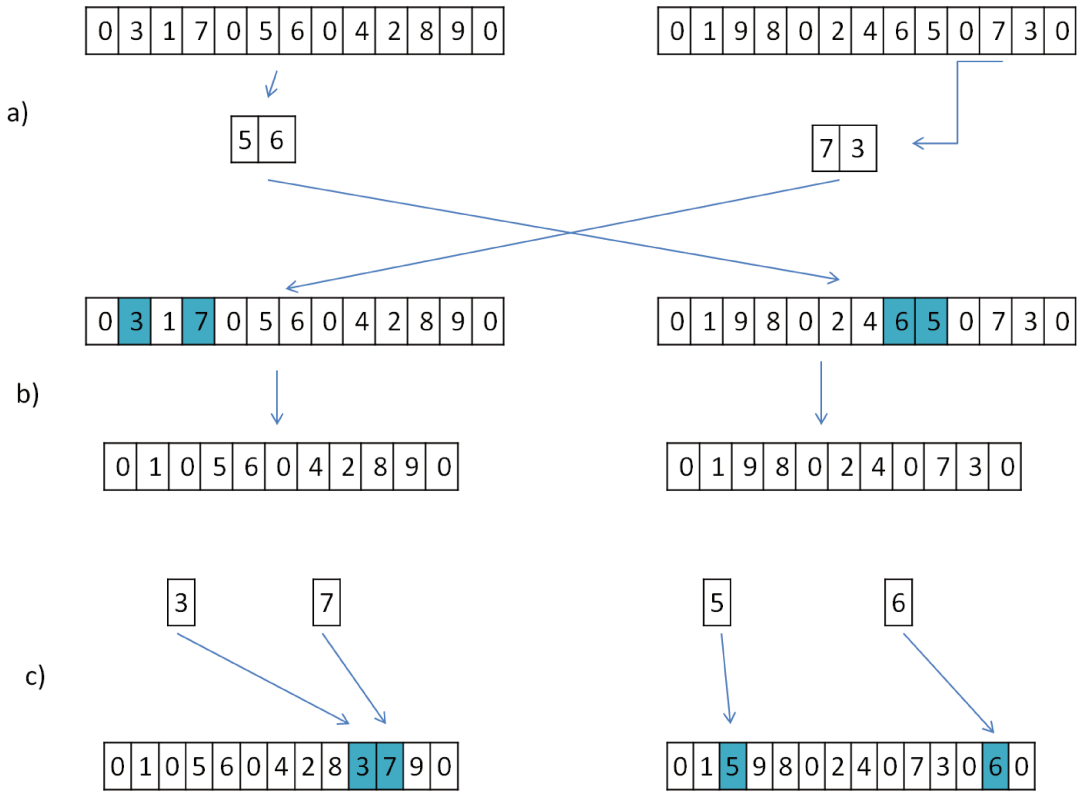


Figura 4.2: Funcionamento do BCR *crossover* padrão. Adaptado de (OMBUKI; ROSS; HANSHAR, 2006).

rotas. Caso necessário, o vértice é inserido em uma nova rota (Figura 4.2 c);.

Como ilustrado na Figura 4.2, no primeiro passo uma rota de cada pai é selecionada. No segundo passo é realizada a remoção de vértices e em seguida, no terceiro passo, os vértices são reinseridos.

A implementação utilizada difere levemente do BCR padrão, pois após a remoção dos vértices é feita uma nova demarcação de rotas.

O OX crossover foi desenvolvido originalmente para TSP e foi aplicado neste trabalho com uma adaptação para lidar com rotas ao invés de cortes. Para cada par de soluções pai, os seguintes passos são tomados.

1. Uma rota é selecionada aleatoriamente de um dos pais, pai A (esquerda) (Figura 4.3 a);

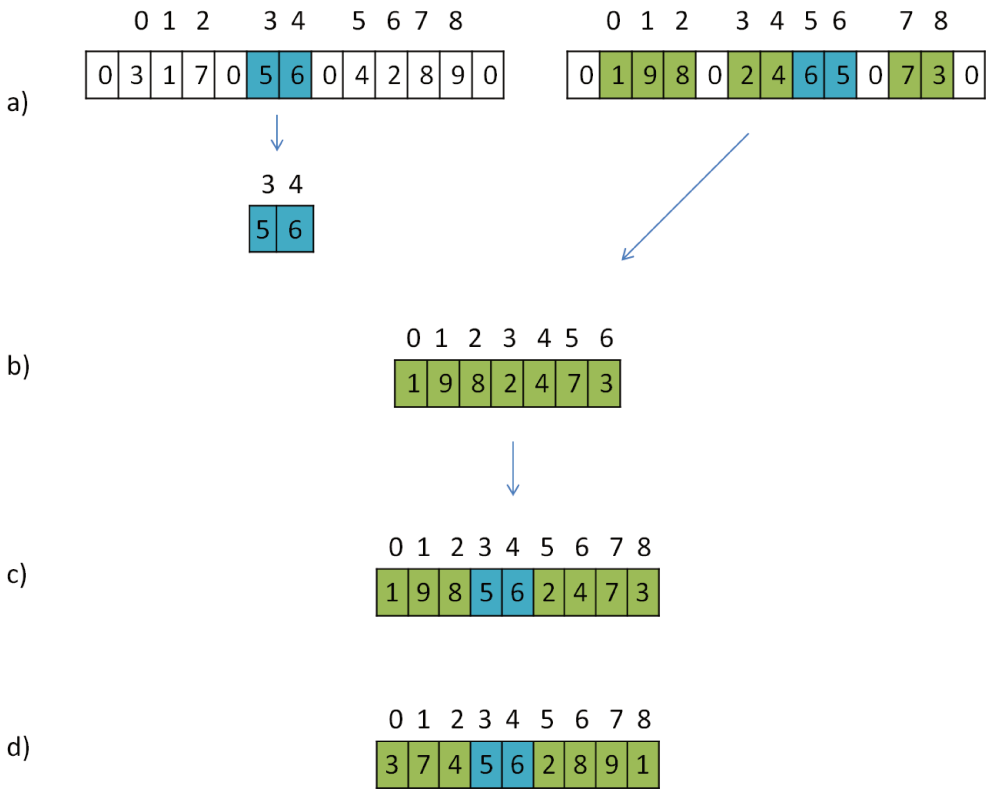


Figura 4.3: Funcionamento do OX *crossover*. Fonte: própria.

2. Um filho sem delimitação de rotas é criado inicialmente apenas com os vértices do pai B (direita), em sua ordem original, excluindo os vértices presentes na rota selecionada do pai A (esquerda) (Figura 4.3 b);

3. Os vértices da rota selecionada do pai A (esquerda) são inseridos no filho a partir da mesma posição em que estariam no pai A (esquerda) (Figura 4.3 c);

4. Um segundo filho é criado da mesma maneira, mas com os vértices do pai B (direita) em ordem inversa (Figura 4.3 d).

Para a etapa de mutação, três vértices aleatórios são selecionados da sequência de inteiros sem demarcação de rotas. Estes vértices são trocados de posição entre si e é feita uma nova demarcação de rotas. A melhor

combinação é usada no cromossomo. Note que a mutação pode piorar o custo da rota.

Finalizando o AG, após a etapa de mutação, a população é atualizada com os x melhores indivíduos sobrevivendo para a próxima geração, sendo x o tamanho definido previamente.

A função objetivo usada no cálculo de *fitness* para seleção dos melhores indivíduos funciona da seguinte forma. Primeiro é considerado o número de veículos da solução. Em caso de empate, é considerado o comprimento da menor rota. Critérios de parada: 100 gerações ou 20 gerações sem melhoria.

Além do AG, dois algoritmos incrementais de troca de arestas foram utilizados. O primeiro algoritmo foi o 2-opt (LIN, 1965). Trata-se de um processo determinístico que testa, para cada rota de uma solução, todas as trocas de pares de arestas possíveis e realiza a melhor delas até que o custo da rota não possa mais melhorar.

O segundo algoritmo é uma variação de *String Exchange* (BREEDAM, 1994), o qual executa trocas de sequências de vértices entre rotas de uma solução da seguinte forma.

1. Duas rotas são selecionadas aleatoriamente da solução (Figura 4.4 a);
2. Um seguimento de tamanho pré definido n é extraído de uma posição aleatória de cada rota (Figura 4.4 b);
3. O seguimento extraído de uma rota é trocado pelo seguimento da outra rota (Figura 4.4 c);
4. Caso haja melhora da solução, a troca é mantida e os passos anteriores se repetem x vezes;

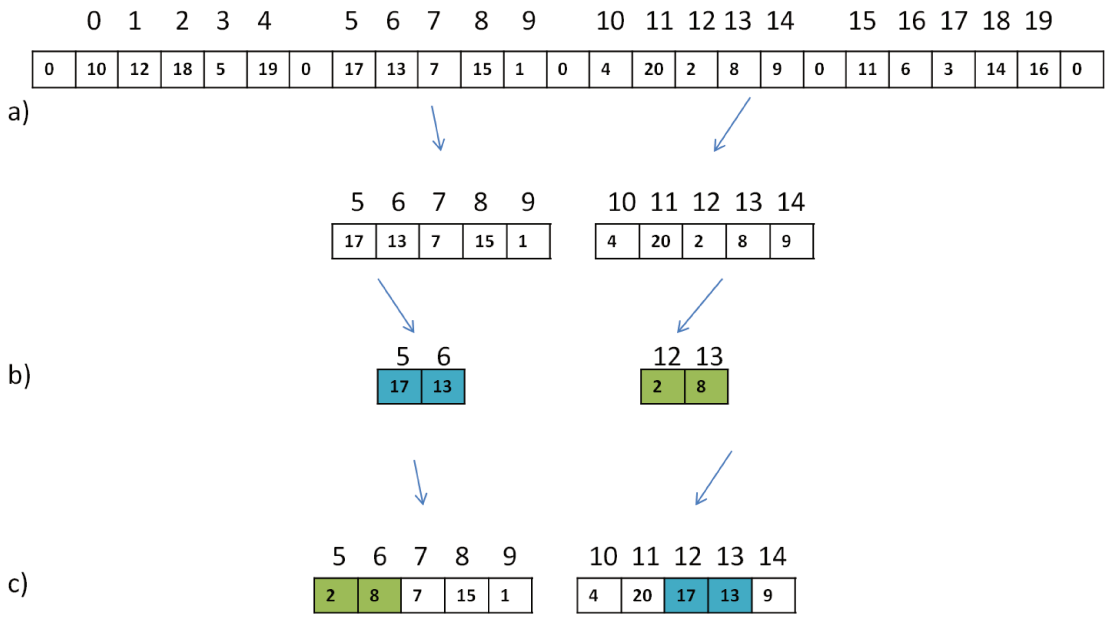


Figura 4.4: Funcionamento do *String Exchange*. Fonte: própria.

5. Após x iterações, todo o processo recomeça com $n \leftarrow n - 1$, até que $n = 0$.

Os valores iniciais de n e x para este algoritmo foram 3 e 30000, respectivamente.

Na parte experimental foram feitas duas variantes de AG. Na primeira, a busca local 2-opt foi aplicada à população resultante ao final de cada geração do AG. Na segunda, não foi aplicado o 2-opt. Em todos os experimentos, o *String Exchange* é aplicado ao melhor indivíduo produzido pelo AG.

4.2 Instâncias Utilizadas

Quatro conjuntos de instâncias foram utilizados neste trabalho. Os conjuntos RealWorldVRP, ManhattanPilot e o example obtidos em (MEIRA; MARTINS; MENZORI; ZENI, 2018; ZENI; MENZORI; MARTINS; MEIRA, 2016) para o PostVRP e o conjunto X (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017) para CVRP.

O conjunto para PostVRP foi desenvolvido a partir de um estudo de caso envolvendo os correios da cidade de Artur Nogueira onde em torno de 25 carteiros realizam até 30.000 entregas por dia. Neste problema os carteiros a pé são os veículos e o comprimento das rotas é medido em unidades de tempo que equivalem ao deslocamento e entrega da correspondência pelos carteiros de um ponto a outro. Cada instância modela um dia de trabalho dos carteiros cujo expediente corresponde a 6-8 horas de trabalho, o que limita o comprimento das rotas.

Adicionalmente, os carteiros não possuem capacidade máxima de carga, uma vez que existe um veículo de apoio que os reabastece quando necessário. O problema proposto é multi-objetivo, desejando-se minimizar o comprimento médio das rotas, minimizar o número de carteiros e, finalmente, minimizar a variação dos comprimentos das rotas. Para este trabalho, porém, a função objetivo foi simplificada para focar na minimização do número de veículos, seguida pelo somatório das rotas.

Como mostra a Tabela 4.2, temos 113 instâncias para PostVRP. As 78 instâncias do conjunto RealWorldVRP estão divididas em quatro grupos. O grupo *Toy* possui instâncias menores, de até 5.000 pontos de entrega, para validação de algoritmos. O *Normal* possui casos mais realistas, de 10.000 a 14.000 pontos. Por fim, os grupos *OnStrike* e *Christmas* representam situações mais atípicas, como feriados de fim de ano, chegando a 30.000 pontos de entrega.

Adicionalmente, a ferramenta usada para gerar as instâncias dos correios também oferece os conjuntos *example* e *ManhattanPilot*, de até 10.000 pontos.

Grupo	N. de Instâncias	Tamanho (min./max.)
example	5	0/10000
Manhattan Pilot	30	3/5000
RealWorldPost Toy	30	3/5000
RealWorldPost Normal	15	10000/14000
RealWorldPost OnStrike	15	15000/19000
RealWorldPost Christmas	18	20000/30000

Tabela 4.2: Instâncias PostVRP.

O conjunto X foi proposto em 2017 para complementar os já consolidados conjuntos do CVRP clássico. Estas instâncias, que podem ter até 1.000 pontos de entrega, visam oferecer uma diversidade maior de problemas, promover o desenvolvimento de métodos mais flexíveis e permitir sua extensão para uso em outras variantes de VRP. Neste trabalho, tal conjunto foi usado principalmente por contar com ótimos conhecidos, proporcionando uma melhor avaliação dos métodos propostos.

Capítulo 5

Experimentos Computacionais

Todos os experimentos foram executados em um *cluster* IBM com 2.6-3.4GHz e 768GB de memória (processo FAPESP projeto 2010/50646-6). Foram utilizadas 24 threads e 32GB de RAM para cada processo.

5.1 Testes Preliminares

Testes preliminares foram realizados no *cluster* citado anteriormente para a definição manual de parâmetros do AG. Para cada teste, o parâmetro sendo testado teve seu valor variado entre algumas opções escolhidas de forma arbitrária enquanto os demais parâmetros foram fixados. No primeiro teste todos os demais parâmetros foram fixados em valores também escolhidos de forma arbitrária. A partir do segundo teste os parâmetros já testados foram fixados no melhor valor obtido pelos testes anteriores.

Primeiro teste: Número de *threads*. Para uma execução mais rápida do AG, a etapa de *crossover* foi implementada fazendo uso de múltiplas *threads*. O primeiro teste consistiu em executar algumas instâncias do conjunto `example` variando o número de *threads* enquanto o tamanho da população, a taxa de *crossover* e a taxa de mutação foram mantidos em 50, 0.95 e 0.05,

respectivamente. O acréscimo de *threads* mostrou uma redução na duração total do AG até 24 *threads*. A partir deste experimento, foi definido o número de *threads* como 24.

Segundo teste: Tamanho da população. O segundo teste teve como objetivo definir o melhor tamanho de população de modo que a quantidade de indivíduos fosse grande o bastante para proporcionar melhores e mais variadas soluções, mas ainda pequena o suficiente para não inflar demais o tempo de execução do AG.

Utilizando a instância `RealWorldPostToy-200-0` de tamanho 200 do conjunto *RealWorld*, quatro tamanhos de população foram testados: 25, 50, 75 e 100. *Threads*, taxa de *crossover* e taxa de mutação foram mantidos em 24, 0.95 e 0.05, respectivamente. A instância foi executada 40 vezes para cada tamanho de população, de acordo com as configurações de experimento detalhadas na Seção 5.2.

Os resultados indicaram qualidade parecida para todos os tamanhos de população, porém os tempos de execução foram 7, 51, 104 e 191 minutos para populações de tamanho 25, 50, 75 e 100, respectivamente. O tamanho de população escolhido para os experimentos foi de 50 por proporcionar um bom equilíbrio entre resultados e velocidade.

Terceiro teste: Taxa de mutação. O valor da mutação foi testado para a mesma instância `RealWorldPostToy-200-0` com número de *threads*, tamanho de população e taxa de *crossover* iguais a 24, 50 e 0.95, respectivamente. Foram testados os valores 5%, 10% e 15%. O melhor resultado foi obtido para taxa de mutação igual a 10%.

Ao final dos testes preliminares, os valores para número de *threads*, tamanho de população e taxa de *mutação* foram definidos como 24, 50 e 0.10, respectivamente, para todos os experimentos seguintes.

5.2 Experimentos

O principal experimento desta pesquisa envolveu comparar o desempenho do AG usando dois operadores diferentes de *crossover* e o uso ou não uso de busca local. Foram então realizadas quatro configurações de experimentos de AG:

(a) Utilizando o operador BCR-Crossover com algoritmo 2-opt para rotas individuais aplicado ao final de cada geração para toda a população (BCR LS *Enabled*);

(b) Utilizando o operador BCR-Crossover sem busca local 2-opt (BCR LS *Disabled*);

(c) Utilizando o operador OX-Crossover com algoritmo 2-opt para rotas individuais aplicado ao final de cada geração para toda a população (OX LS *Enabled*).

(d) Utilizando o operador OX-Crossover sem busca local 2-opt (OX LS *Disabled*);

Para todos os casos, ao final do AG, é feita a busca local *String Exchange* na melhor solução encontrada.

Cada instância foi executada dez vezes em cada uma das quatro configurações, totalizando quarenta execuções por instância.

5.3 Instâncias do PostVRP

Nos experimentos descritos a seguir, utilizamos as instâncias descritas em (MEIRA; MARTINS; MENZORI; ZENI, 2018). Tais instâncias são uma versão multi-objetivo do VRP com comprimento máximo da rota limitado. Neste benchmark, o nome da instância é composta por três partes. Por exemplo, a instância `ManhattanPilot_500_2` pertence ao grupo ManhattanPilot, possui 500 pontos de entrega e possui identificador $ID = 2$. De 113 instâncias no total, foram utilizadas apenas as 64 menores neste trabalho devido ao alto tempo de execução dos casos maiores. A Figura 5.1 ilustra que a complexidade dos algoritmos propostos se aproxima a $O(n^{1.5})$, o que torna proibitivo realizar 40 execuções de experimento para as instâncias com 10.000 clientes ou mais.

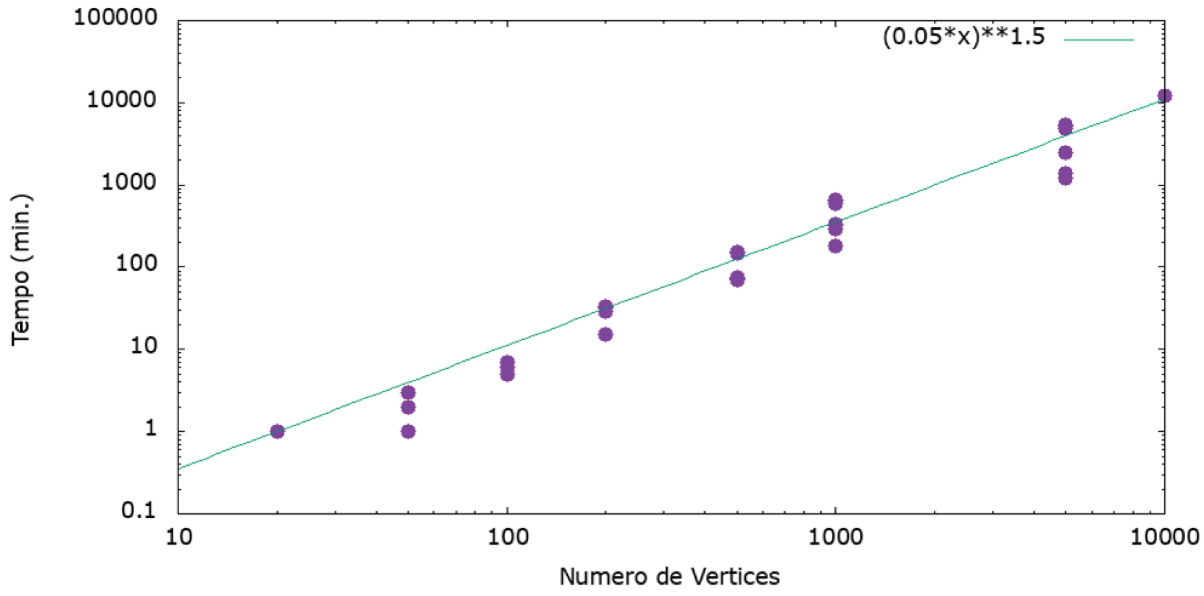


Figura 5.1: Relação entre o tempo de 40 execuções dos algoritmos propostos e o tamanho das instâncias PostVRP utilizadas.

Nas Figuras 5.2 e 5.3 comparamos o operador OX LS *Enabled* com OX LS *Disabled* para uma instância de tamanho 100 e para outra de tamanho

10.000. A Figura 5.2 contém a distância *versus* geração e a Figura 5.3 contém o número de veículos *versus* geração. Em ambos os casos, o operador de busca local trouxe uma convergência mais acentuada, criando uma grande diferença no valor das soluções obtidas já a partir da segunda geração do AG. Esta convergência levou ao cumprimento mais rápido da condição de parada "20 gerações sem melhora da solução". Observando a distância entre as curvas para o caso com a instância de tamanho 100 em relação ao caso com tamanho 10.000, também fica evidente que esta convergência é ainda mais significativa para a instância maior.

Nas Figuras 5.4 e 5.5 comparamos os operadores BCR e OX nas configurações LS *Enabled* e LS *Disabled* para uma instância de tamanho 10.000. A Figura 5.4 contém a distância *versus* geração e a Figura 5.5 contém o número de veículos *versus* geração. Na Figura 5.5, o mínimo número de veículos para (OX LS *Enabled*, OX LS *Disabled*) foi $(82, 799)$ respectivamente. Já para (BCR LS *Enabled*, BCR LS *Disabled*), o número mínimo de veículos foi $(74, 701)$. Podemos observar que, assim como descrito no parágrafo anterior, o uso de busca local levou a uma melhora da convergência do algoritmo. Tal melhora ocorreu para ambos os operadores, sendo que o operador BCR teve um desempenho superior ao OX.

Em todos os experimentos utilizamos dois critérios de parada: 100 gerações totais ou 20 gerações sem melhoria. Em algumas situações específicas, a convergência acelerada, observada anteriormente nas configurações LS *Enabled*, fez com que o AG prematuramente atingisse 20 gerações sem melhora. Enquanto isso, a configuração LS *Disabled*, apresentando uma convergência mais longa e suave, obteve resultado levemente superior, como ilustrado nas Figuras 5.6 e 5.7.

Comparando os operadores de *crossover*, o operador BCR se mostrou superior ao OX na grande maioria dos casos. Apesar disso, as Figuras 5.8 e 5.9 mostram que com o auxílio do 2-opt aplicado a cada geração, o desempenho do operador OX consegue se aproximar mais do operador BCR. Para instâncias maiores, esta aproximação também parece crescer, como ilustrado nas Figuras 5.10 e 5.11.

Por ser mais simples, o AG com operador OX roda em torno de quatro vezes mais rápido que o AG com operador BCR. Assim, o operador OX continua sendo uma opção competitiva, especialmente se combinado à busca local.

De um total de 113 instâncias descritas em (MEIRA; MARTINS; MENZORI; ZENI, 2018) selecionamos as 62 menores. A Tabela 5.1 contém a solução com mínimo $(k, comprimento)$ encontrada nas 40 execuções, 10 para cada configuração distinta. Foi comparado em primeiro lugar o valor k. Para duas soluções com mínimo k, foi comparado então a soma dos comprimentos das rotas. Além dos custos, a tabela mostra a configuração ou configurações que obtiveram o melhor resultado para cada instância. Houve um empate dos quatro algoritmos para todas as instâncias com até 10 pontos de entrega. Tais instâncias são muito pequenas. Para as demais instâncias, com 11 ou mais pontos de entrega, houve sempre uma configuração vitoriosa. Tal configuração foi: *(a)* BCR LS *Enabled* em 27 casos, *(b)* BCR LS Disable em 13 casos e *(c)* OB LS *Enabled* em 4 casos.

As instâncias com 500 pontos de entrega gastaram em torno de 2 horas de execução. Instâncias com 5.000 pontos de entrega gastaram, no máximo, 93 horas. Cada execução rodou com até 24 *threads* em paralelo. Observe que

o tempo diz respeito a soma das 40 execuções e foi medido ignorando-se o paralelismo.

Em relação a variância de comprimento entre as rotas de uma solução, em alguns casos a diferença de tamanho entre a menor e a maior rota ficou em torno de 10%, porem em outros casos essa diferença foi bem superior, com uma rota chegando a ser 11 vezes maior do que outra. Tal comportamento foi provavelmente fruto do critério de *fitness* do AG, que favorece soluções que tenham uma de suas rotas pequena, mas sem considerar o tamanho das demais. Outro fator responsável foi a abordagem gulosa empregada na demarcação de rotas durante a criação de um cromossomo: uma nova rota só é criada após a anterior estar cheia, permitindo que a última rota seja desproporcionalmente menor que as outras.

A instância `RealWorldPostNormal-10000-0`, não presente na Tabela 5.1, foi executada apenas 16 vezes, 10 para BCR LS *Disabled* e 6 para BCR LS *Enabled*, devido a seu alto tempo de execução, que somou 15.137 minutos para estas execuções. Nestas circunstâncias, a configuração BCR LS *Enabled* obteve o melhor resultado com 32 veículos e 688.541.690.067 microssegundos de comprimento.

As Figuras 5.12 e 5.13 ilustram uma das soluções obtidas sobre o mapa da cidade de Artur Nogueira. Note que estes grafos estão apenas ligando os vértices em linha reta de acordo com a solução obtida, o que não corresponde ao real trajeto dos carteiros, já que ignoram o *layout* das ruas.

5.4 Instâncias do Conjunto X

Todas as instâncias utilizadas até o momento não possuem limitantes conhecidos. Sendo assim, este trabalho também realizou o mesmo experimento

utilizando as instâncias do "Conjunto X" (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017), as quais possuem ótimo conhecido. As instâncias deste conjunto são do CVRP.

Neste *benchmark*, o nome da instância possui o número de entregas e o número de veículos. Por exemplo, a instância `X-n101-k25` possui 101 pontos de entregas e 25 veículos.

As Tabelas 5.2, 5.3 e 5.4 mostram a configuração que obteve o melhor resultado de k e comprimento para cada instância, assim como seus valores ótimos. Para este conjunto as configurações que obtiveram o melhor desempenho foram: *(a)* BCR LS *Enabled* em 57 casos, *(c)* OX LS *Enabled* em 38 casos e *(b)* BCR LS *Disabled* em 5 casos.

A Figura 5.14 compara o k obtido em relação ao ótimo para cada uma das 100 instâncias do conjunto X. Em 46% dos casos o algoritmo obteve o número de rotas ótimo. Em 91% das instâncias, o k obtido ficou até 10% maior que o k ótimo.

Considerando o comprimento, devido aos critérios de *fitness* do AG estarem focados na minimização do k, os resultados obtidos mostraram qualidade variando entre 27% e quase 200% acima do ótimo, para as instâncias `X-n190-k8` e `X-n561-k42` respectivamente. A qualidade média das melhores soluções foi em torno de 90% acima do ótimo.

O maior tempo de execução foi de 68 minutos e o tempo médio de execução foi de 20 minutos.

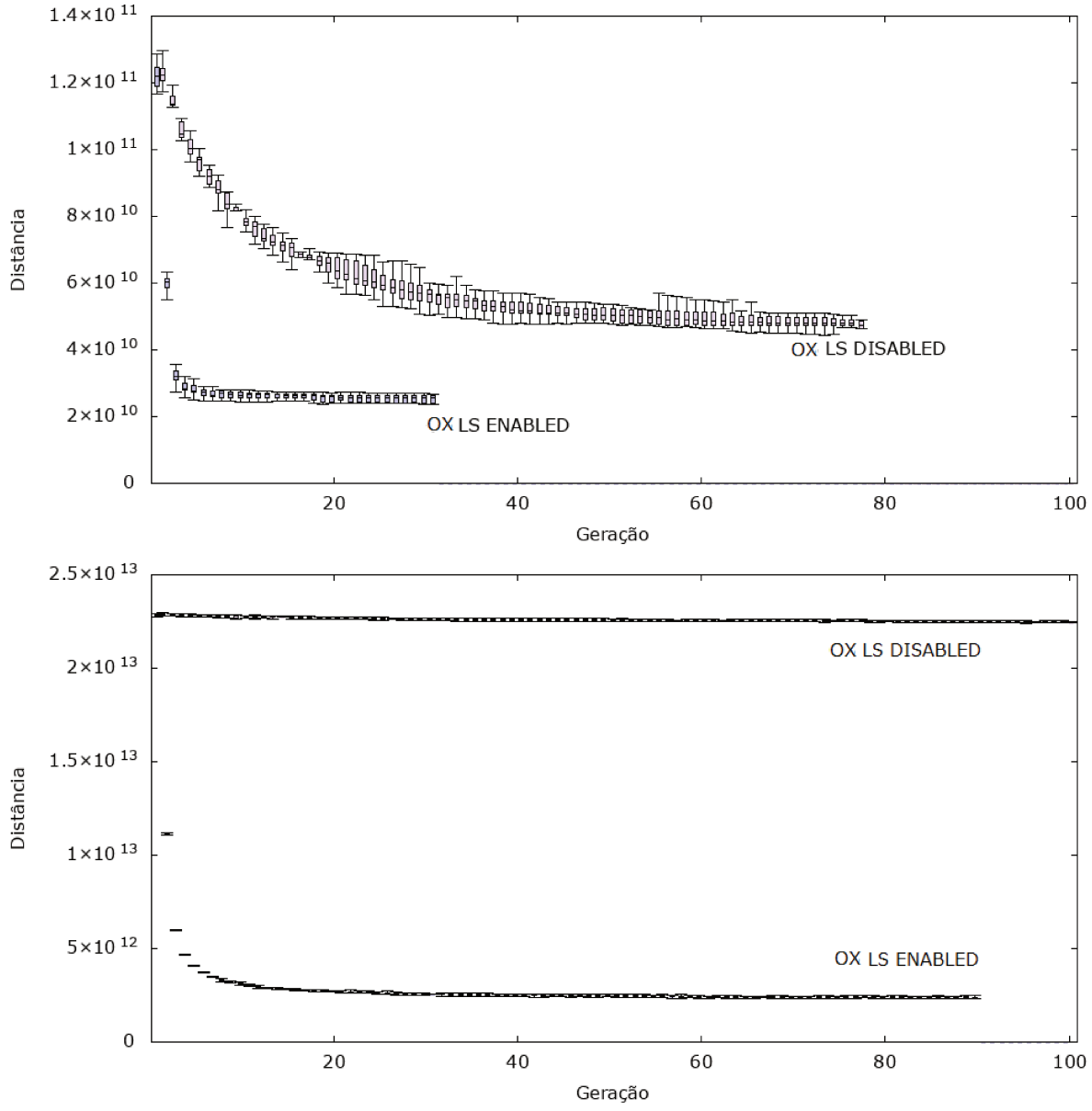


Figura 5.2: Distância versus geração para as instâncias `RealWorldPostToy-100-1` (acima) e `example-10000-5` (abaixo), usando o operador OB com e sem busca local ao final de cada geração.

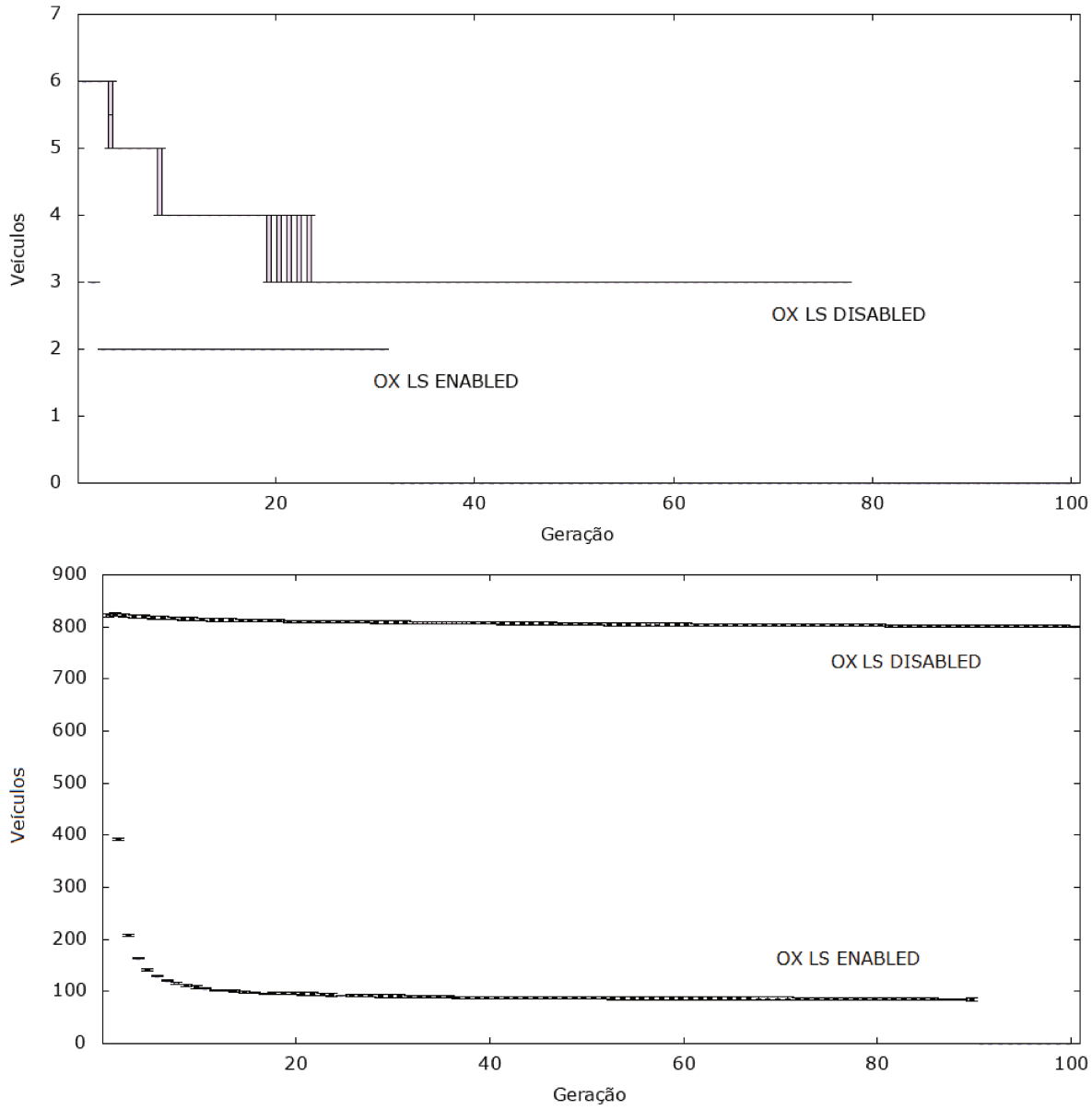


Figura 5.3: Número de veículos versus geração para as instâncias `RealWorldPostToy-100-1` (acima) e `example-10000-5` (abaixo), usando o operador OB com e sem busca local ao final de cada geração.

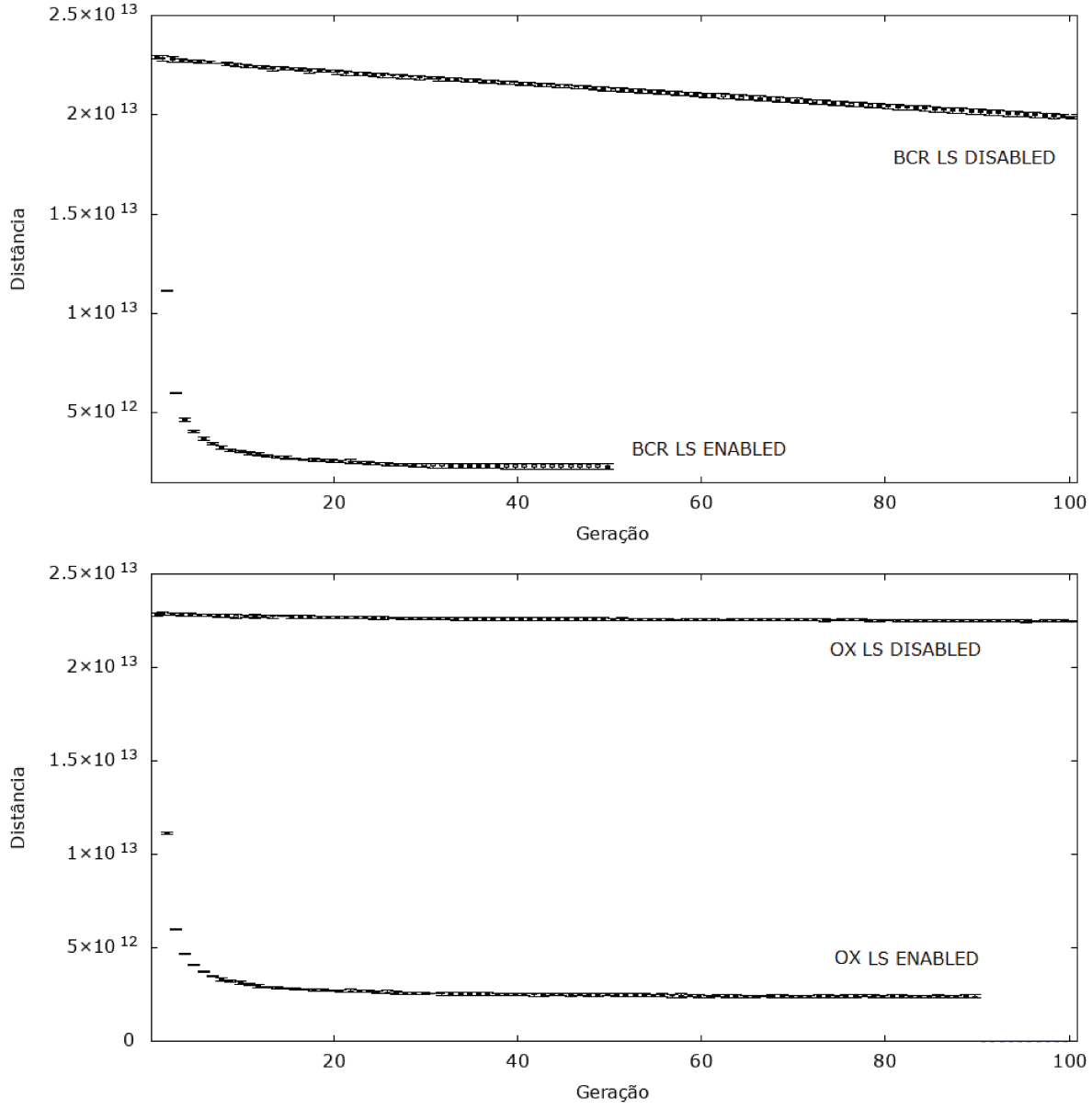


Figura 5.4: Distância versus geração para os operadores BCR (acima) e OX (abaixo), instância `example-10000-5`, com e sem busca local ao final de cada geração.

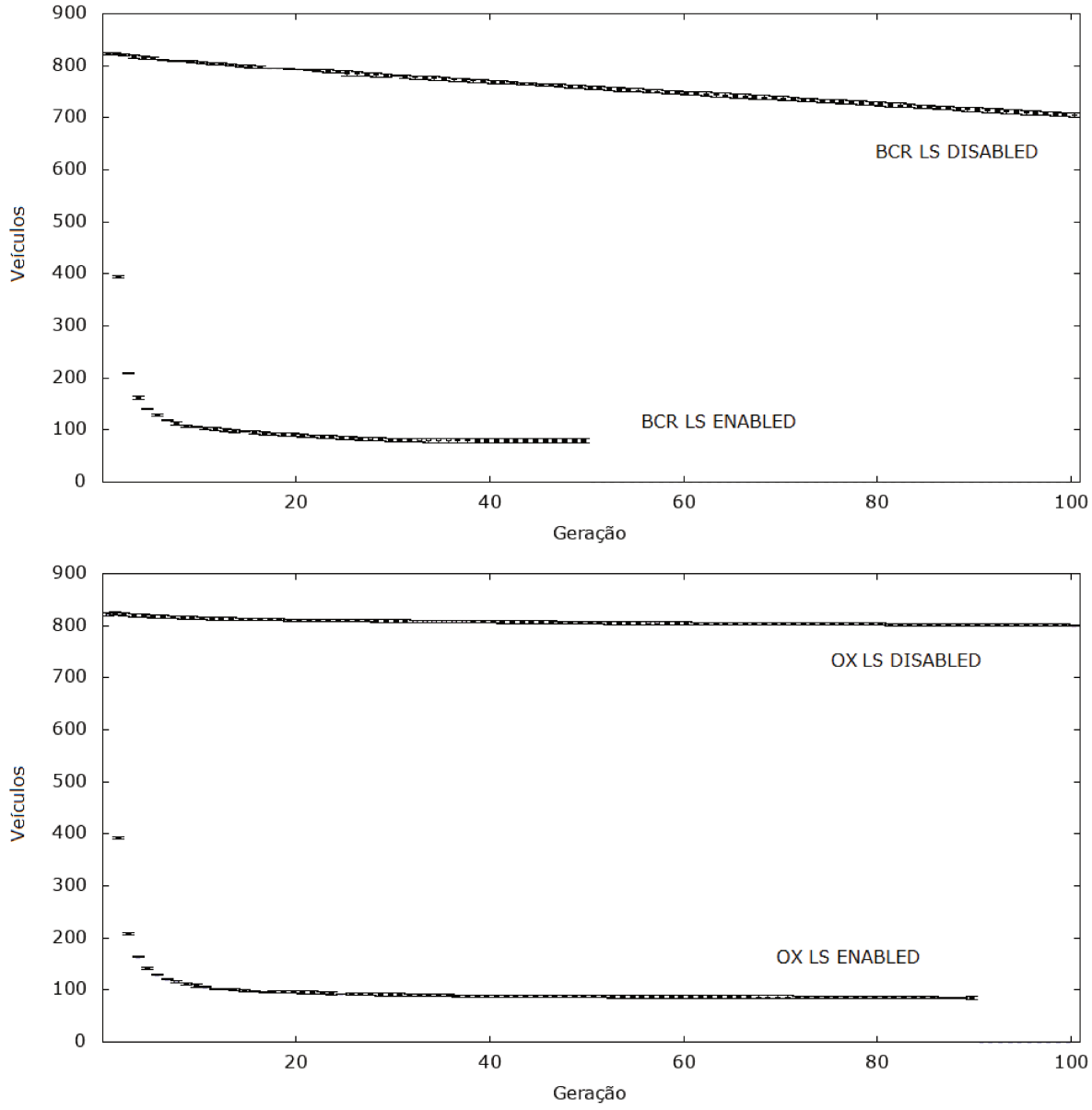


Figura 5.5: Número de veículos versus geração para os operadores BCR (acima) e OX (abaixo), instância `example-10000-5`, com e sem busca local ao final de cada geração.

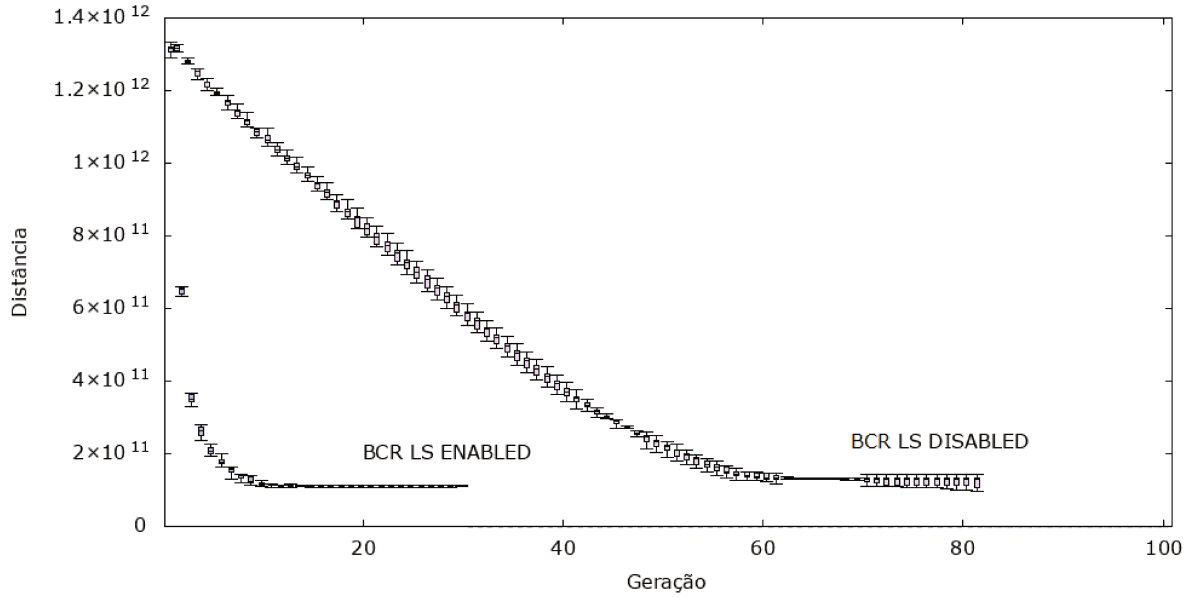


Figura 5.6: Distância versus geração usando o operador BCR, instância `RealWorldPostToy-1000-2` com e sem busca local ao final de cada geração.

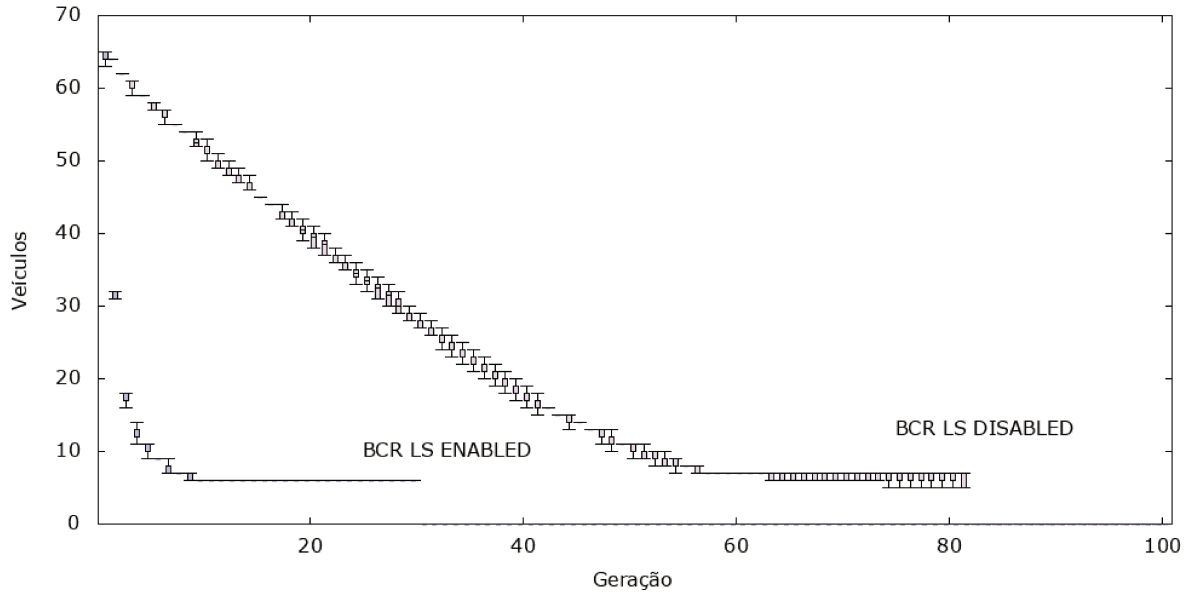


Figura 5.7: Número de veículos versus geração usando o operador BCR, instância `RealWorldPostToy-1000-2` com e sem busca local ao final de cada geração.

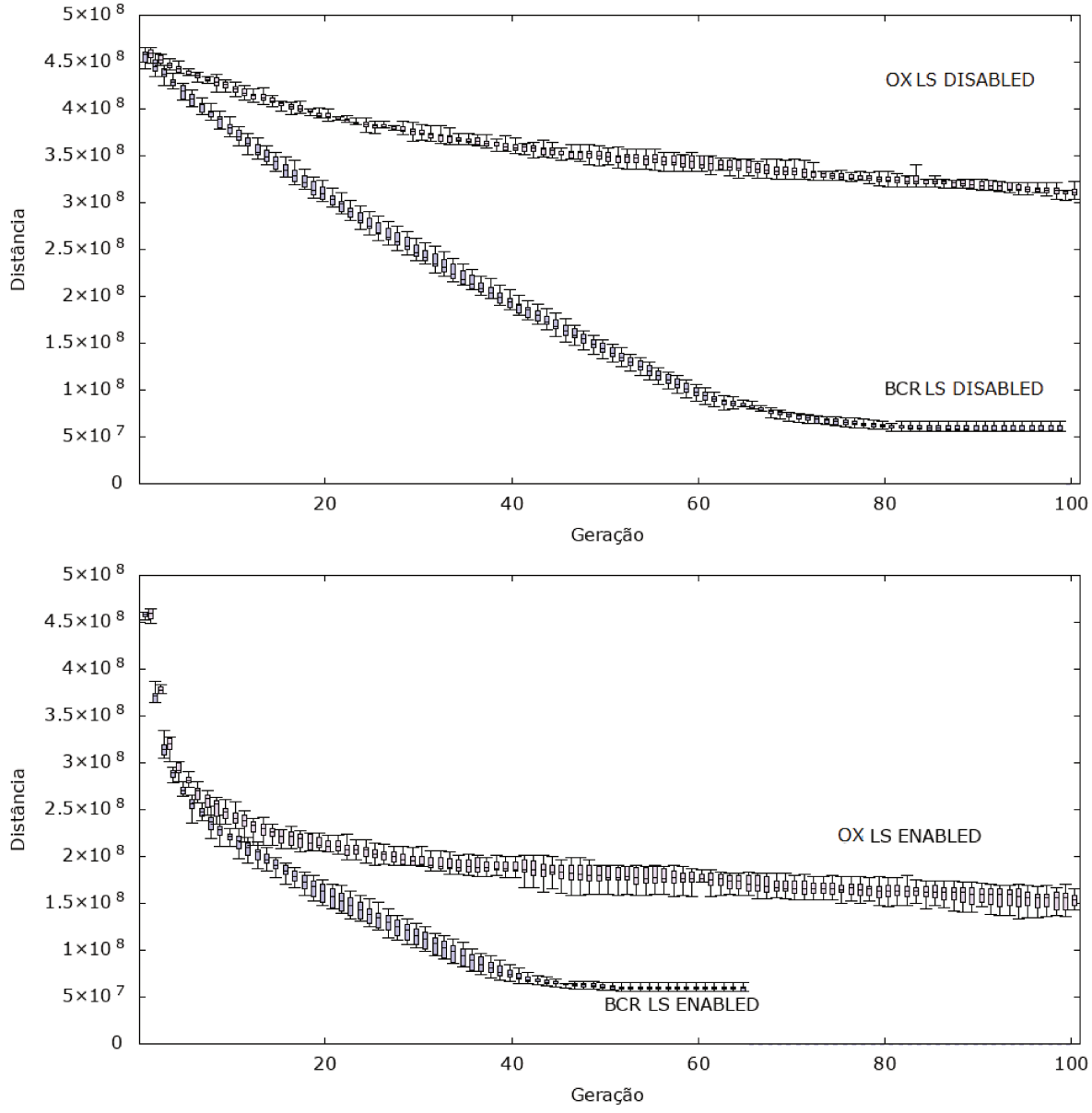


Figura 5.8: Distância versus geração para os operadores BCR e OX, instância `ManhattanPilot-500-1`, sem busca local (acima) e com busca local (abaixo) ao final de cada geração.

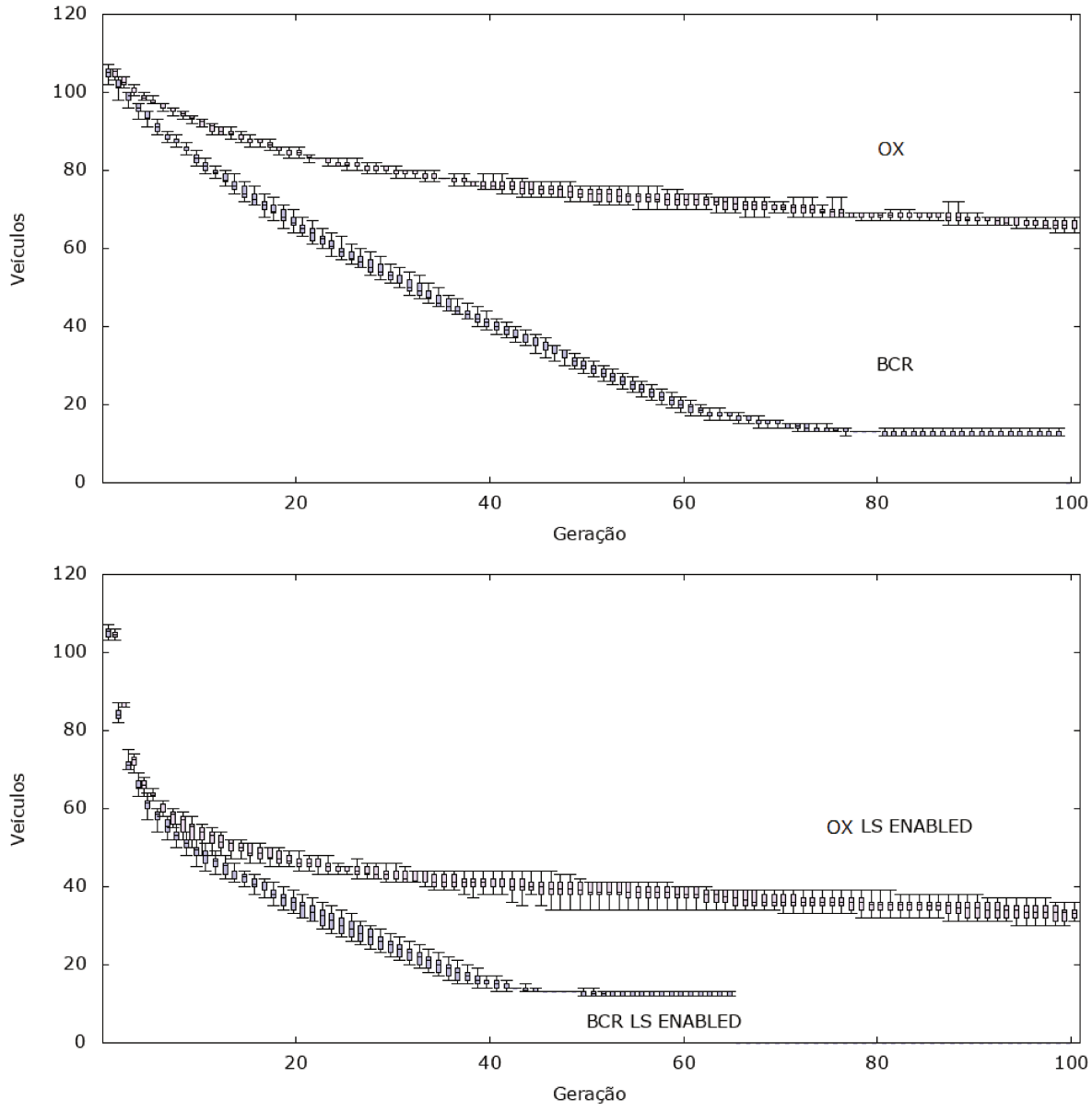


Figura 5.9: Número de veículos versus geração para os operadores BCR e OX, instância `ManhattanPilot-500-1`, sem busca local (acima) e com busca local (abaixo) ao final de cada geração.

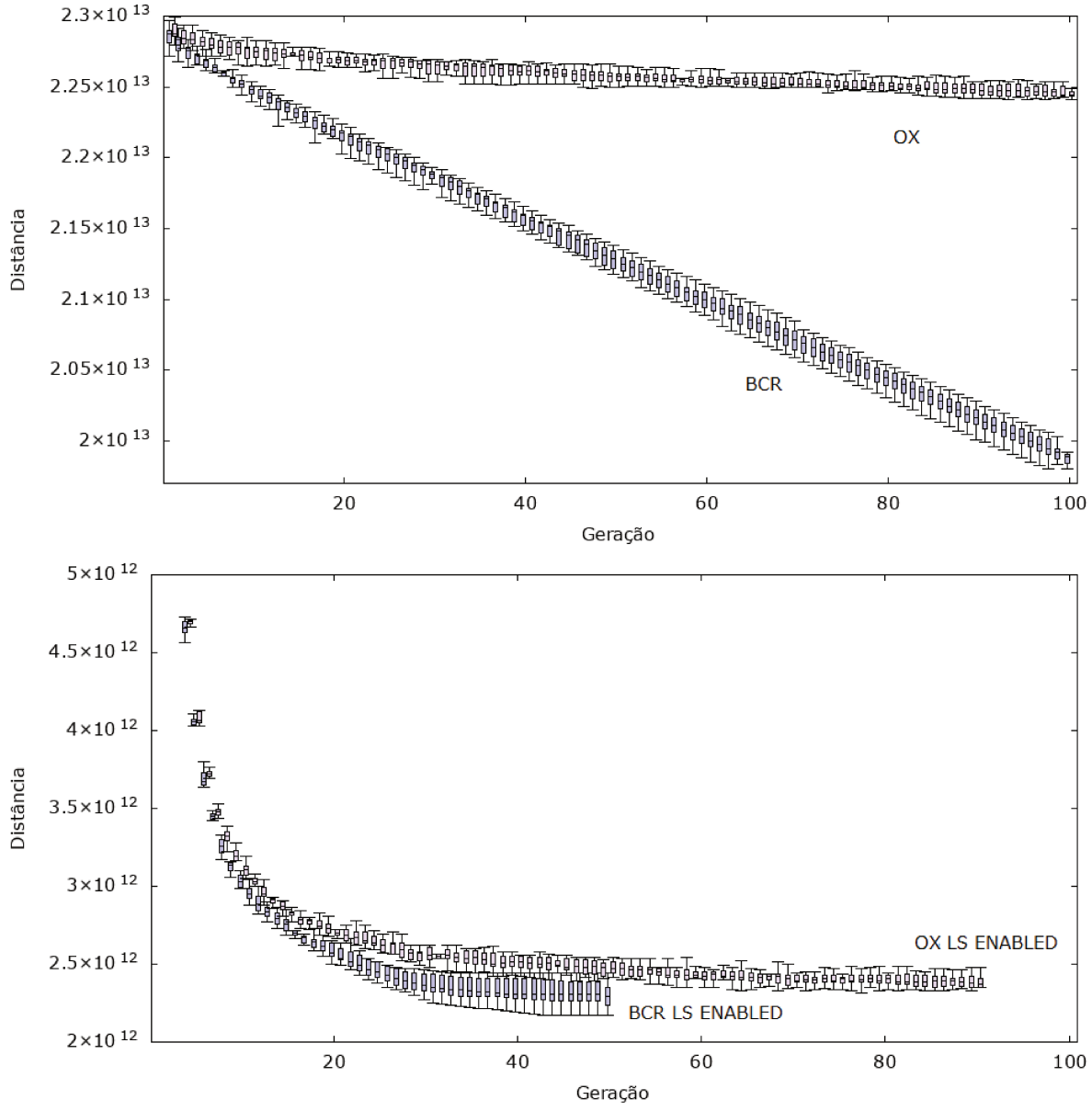


Figura 5.10: Distância versus geração para os operadores BCR e OX, instância `example-10000-5`, sem busca local (acima) e com busca local (abaixo) ao final de cada geração.

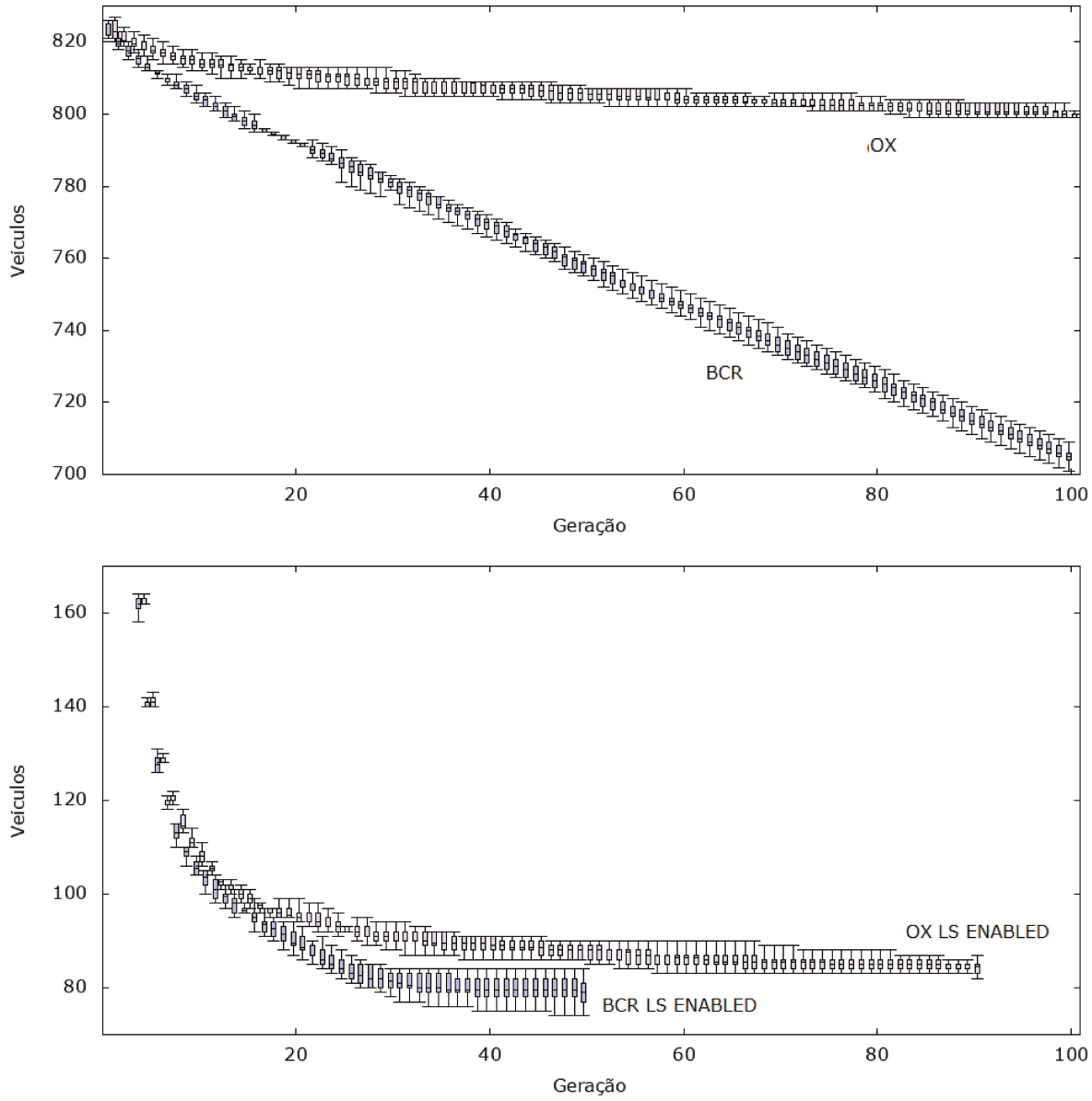


Figura 5.11: Número de veículos versus geração para os operadores BCR e OX, instância `example-10000-5`, sem busca local (acima) e com busca local (abaixo) ao final de cada geração.

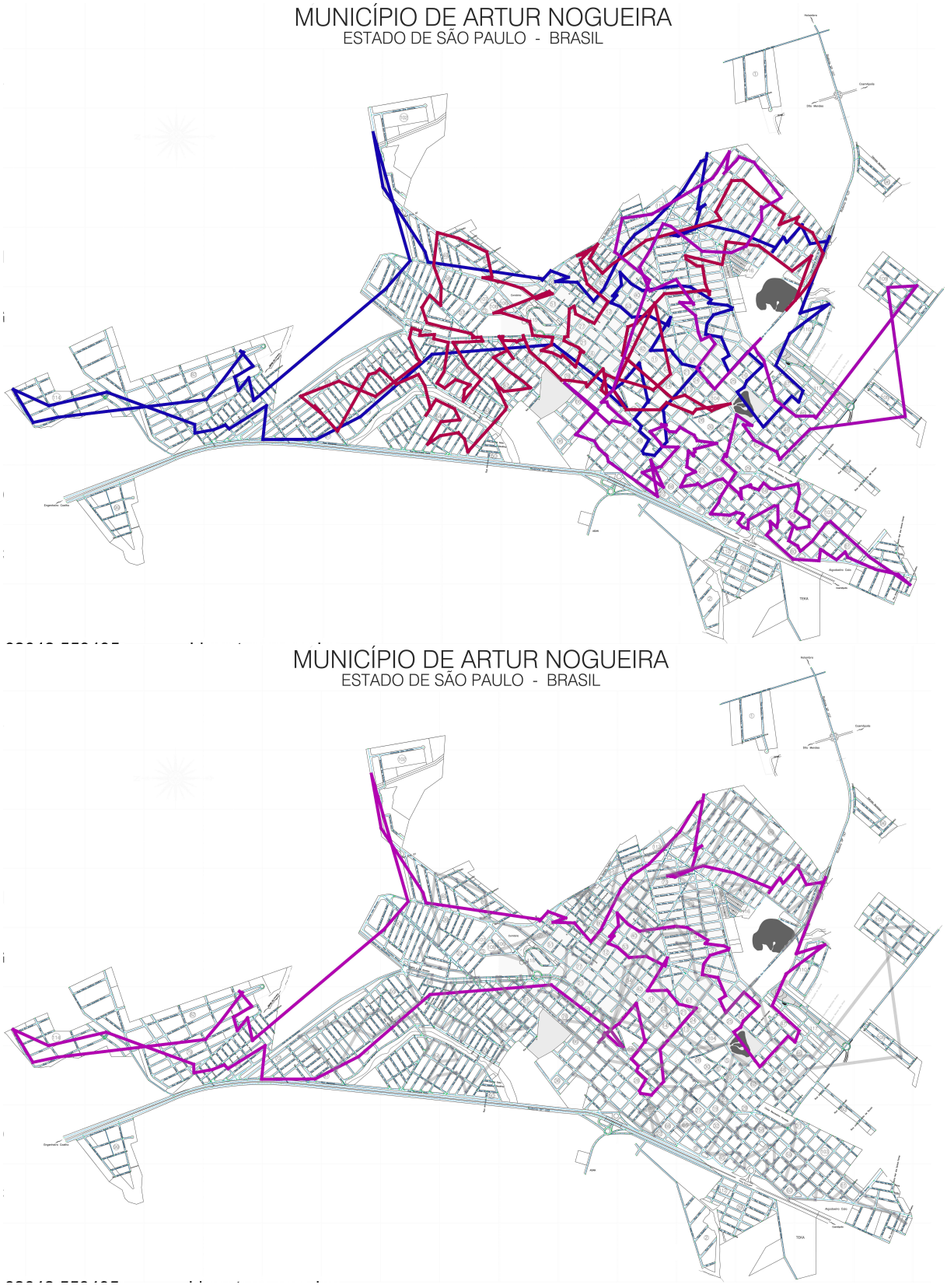


Figura 5.12: Mapa de Artur Nogueira com uma solução da instância `RealWorldPostToy-500-0` desenhada sobre ele, parte 1. A primeira imagem mostra a solução completa enquanto a segunda mostra uma rota isolada. Grafo não representativo do real trajeto do carteiro.

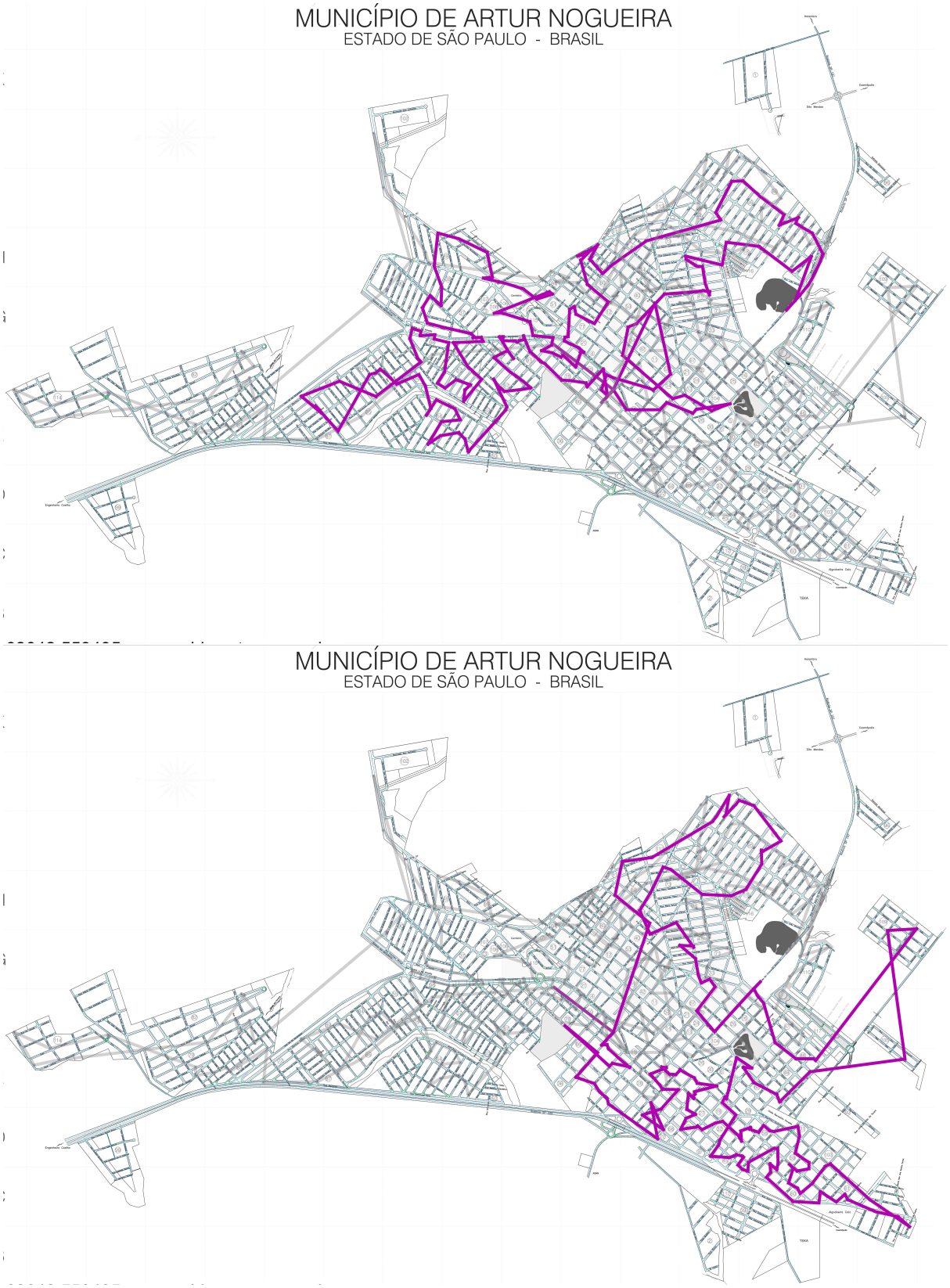


Figura 5.13: Mapa de Artur Nogueira com uma solução da instância `RealWorldPostToy-500-0` desenhada sobre ele, parte 2. Ambas as imagens mostram uma rota isolada da solução. Grafo não representativo do real trajeto do carteiro.

Tabela 5.1: Melhores resultados obtidos por instância. Seja (BCR LS *Enabled*, BCR LS *Disabled*, OB LS *Enabled*, OB LS *Disabled*) igual a (A, B, C, D) respectivamente. Mostramos na tabela o melhor $(k, comprimento)$ obtidos, o comprimento médio das 10 execuções do melhor algoritmo, e o tempo das 40 execuções somadas. Também destacamos qual algoritmo encontrou a melhor solução.

Inst	Melhor alg.	min k	min compr.	tempo (min.)	min compr. médio
exp-10-5	ABCD	1	15095885629	0	15095885629±0
exp-100-5	A	2	35236853041	6	37417116907±1.e^{09}
exp-1000-5	A	8	211470000000	318	265826833218±2.e^{10}
exp-10000-5	A	71	1876720000000	12138	2305001403534±1.e^{11}
ManP-3-0	ABCD	1	3709581	0	3709581±0
ManP-3-1	ABCD	1	2485357	0	2485357±0
ManP-3-2	ABCD	1	3786186	0	3786186±0
ManP-5-0	ABCD	1	4686357	0	4686357±0
ManP-5-1	ABCD	1	2686480	0	2686480±0
ManP-5-2	ABCD	1	4134533	0	4134533±0
ManP-10-0	ABCD	2	6575903	0	6575903±0
ManP-10-1	ABCD	1	4497191	0	4497191±0
ManP-10-2	ABCD	1	4678544	0	4678544±0
ManP-20-0	A	2	7202261	0	7327752±2.e^{05}
ManP-20-1	A	2	6982255	0	7223962±2.e^{05}
ManP-20-2	B	2	8728363	0	8784971±7.e^{04}
ManP-50-0	A	3	12611093	2	13069920±3.e^{05}
ManP-50-1	A	3	10813610	1	11515364±6.e^{05}
ManP-50-2	B	3	11945041	1	12969410±7.e^{05}
ManP-100-0	B	4	16962018	5	18928856±1.e^{06}
ManP-100-1	B	4	16948889	5	19446718±2.e^{06}
ManP-100-2	A	4	15728046	5	18416112±2.e^{06}
ManP-200-0	B	6	26546522	15	29687528±2.e^{06}
ManP-200-1	A	6	26249777	15	29653070±3.e^{06}
ManP-200-2	B	6	26257130	15	30836576±3.e^{06}
ManP-500-0	B	11	49087528	71	57058160±4.e^{06}
ManP-500-1	B	12	52109723	73	55282604±2.e^{06}
ManP-500-2	A	11	48930341	76	54196292±3.e^{06}
ManP-1000-0	A	18	87619991	181	93277784±7.e^{06}
ManP-1000-1	A	17	79051618	296	88720720±4.e^{06}
ManP-1000-2	A	20	88791674	336	95021528±4.e^{06}
ManP-5000-0	C	525	2381307163	1372	1020423424±2.e^{07}
ManP-5000-1	C	511	954191742	1199	971550528±1.e^{07}
ManP-5000-2	C	499	2271821549	2494	959469760±2.e^{07}
RWPToy-3-0	ABCD	1	6023937832	0	6023938048±0
RWPToy-3-1	ABCD	1	6826653003	0	6826653184±0
RWPToy-3-2	ABCD	1	4127370448	0	4127370496±0
RWPToy-5-0	ABCD	1	3742109767	0	3742109696±0
RWPToy-5-1	ABCD	1	3461359665	0	3461359616±0
RWPToy-5-2	ABCD	1	5188914899	0	5188914688±0
RWPToy-10-0	ABCD	1	8103109376	0	8103109632±0
RWPToy-10-1	ABCD	1	8727033996	0	8727033856±0
RWPToy-10-2	ABCD	1	6082601657	0	6082601472±0
RWPToy-20-0	B	1	11229149347	1	11245787136±2.e^{07}
RWPToy-20-1	B	1	11253026628	1	11276482560±3.e^{07}
RWPToy-20-2	B	1	9357553819	1	9357553664±0
RWPToy-50-0	A	1	17762849447	3	17857062912±1.e^{08}
RWPToy-50-1	A	1	13197120948	3	13383871488±1.e^{08}
RWPToy-50-2	C	1	15646200132	3	15891240960±1.e^{08}
RWPToy-100-0	A	2	23439941986	7	24544595968±6.e^{08}
RWPToy-100-1	A	1	21404038732	7	23103369216±1.e^{09}
RWPToy-100-2	A	1	21565328275	7	23006742528±1.e^{09}
RWPToy-200-0	A	2	33590426863	29	38419750912±2.e^{09}
RWPToy-200-1	A	2	33655199969	32	36768653312±2.e^{09}
RWPToy-200-2	A	2	32211406237	33	34312071168±2.e^{09}
RWPToy-500-0	A	3	61231448742	150	65014939648±2.e^{09}
RWPToy-500-1	B	4	62710650345	153	66719155246±4.e^{09}
RWPToy-500-2	A	3	60970050148	154	66650648576±3.e^{09}
RWPToy-1000-0	A	5	99477299207	598	105219227648±7.e^{09}
RWPToy-1000-1	A	5	94550435967	659	105187450880±7.e^{09}
RWPToy-1000-2	B	5	89635383711	644	109733601280±1.e^{10}
RWPToy-5000-0	A	17	365962000000	5481	373737521152±2.e^{10}
RWPToy-5000-1	A	17	367079000000	5296	381341696000±2.e^{10}
RWPToy-5000-2	A	18	387842000000	4938	382466621440±1.e^{10}

Tabela 5.2: Execução do AG nas instâncias do Conjunto X, parte 1. Seja (BCR LS *Enabled*, BCR LS *Disabled*, OB LS *Enabled*, OB LS *Disabled*) igual a (A, B, C, D) respectivamente. Mostramos na tabela os valores de k e de comprimento obtidos, assim como seus valores ótimos (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017). Também destacamos qual algoritmo encontrou a melhor solução.

Inst	melhor alg	k	k opt	min comp.	comp. opt
X-n101-k25	26	25	44063	27591	A
X-n106-k14	14	14	33485	26362	C
X-n110-k13	13	13	27895	14971	C
X-n115-k10	10	10	20217	12747	A
X-n120-k6	6	6	18648	13332	C
X-n125-k30	30	30	73212	55539	A
X-n129-k18	18	18	42814	28940	C
X-n134-k13	13	13	17432	10916	C
X-n139-k10	10	10	23779	13590	C
X-n143-k7	7	7	23852	15700	A
X-n148-k46	49	46	73548	43448	A
X-n153-k22	23	22	31347	21220	A
X-n157-k13	15	13	26592	16876	A
X-n162-k11	11	11	27648	14138	C
X-n167-k10	10	10	31881	20557	C
X-n172-k51	52	51	80033	45607	A
X-n176-k26	26	26	66725	47812	A
X-n181-k23	26	23	40927	25569	A
X-n186-k15	15	15	42553	24145	C
X-n190-k8	8	8	21541	16980	A
X-n195-k51	52	51	90848	44225	B
X-n200-k36	37	36	85924	58578	A
X-n204-k19	19	19	41405	19565	C
X-n209-k16	16	16	50192	30656	C
X-n214-k11	11	11	19759	10856	A
X-n219-k73	109	73	216361	117595	B
X-n223-k34	35	34	82646	40437	A
X-n228-k23	23	23	52311	25742	A
X-n233-k16	17	16	44412	19230	C
X-n237-k14	14	14	43971	27042	C
X-n242-k48	50	48	130689	82751	A
X-n247-k50	51	50	52593	37274	A
X-n251-k28	28	28	66165	38684	C
X-n256-k16	16	16	48710	18839	A

Tabela 5.3: Execução do AG nas instâncias do Conjunto X, parte 2. Seja (BCR LS *Enabled*, BCR LS *Disabled*, OB LS *Enabled*, OB LS *Disabled*) igual a (A, B, C, D) respectivamente. Mostramos na tabela os valores de k e de comprimento obtidos, assim como seus valores ótimos (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017). Também destacamos qual algoritmo encontrou a melhor solução.

Inst	melhor alg	k	k opt	min comp.	comp. opt
X-n261-k13	13	13	45932	26558	C
X-n266-k58	61	58	124987	75478	A
X-n270-k35	36	35	81932	35291	C
X-n275-k28	31	28	40938	21245	A
X-n280-k17	17	17	60017	33503	A
X-n284-k15	15	15	34184	20226	C
X-n289-k60	61	60	151198	95151	A
X-n294-k50	51	50	105773	47161	A
X-n298-k31	32	31	78667	34231	C
X-n303-k21	21	21	48267	21744	C
X-n308-k13	13	13	42815	25859	C
X-n313-k71	73	71	150014	94044	A
X-n317-k53	64	53	148034	78355	A
X-n322-k28	28	28	74558	29834	C
X-n327-k20	20	20	55143	27532	C
X-n331-k15	15	15	51356	31103	C
X-n336-k84	86	84	206640	139135	A
X-n344-k43	44	43	106976	42056	A
X-n351-k40	41	40	55176	25928	C
X-n359-k29	30	29	88406	51505	C
X-n367-k17	17	17	41764	22814	C
X-n376-k94	125	94	263329	147713	A
X-n384-k52	53	52	131994	65943	A
X-n393-k38	39	38	88857	38260	C
X-n401-k29	29	29	94269	66187	C
X-n411-k19	19	19	42051	19718	A
X-n420-k130	140	130	193056	107798	A
X-n429-k61	62	61	152393	65501	A
X-n439-k37	40	37	69976	36395	A
X-n449-k29	29	29	107599	55269	A
X-n459-k26	26	26	51203	24145	C
X-n469-k138	146	138	341041	221909	A
X-n480-k70	72	70	171174	89458	A

Tabela 5.4: Execução do AG nas instâncias do Conjunto X, parte 3. Seja (BCR LS *Enabled*, BCR LS *Disabled*, OB LS *Enabled*, OB LS *Disabled*) igual a (A, B, C, D) respectivamente. Mostramos na tabela os valores de k e de comprimento obtidos, assim como seus valores ótimos (UCHOA; PECIN, D.; PESSOA; POGGI; VIDAL; SUBRAMANIAN, 2017). Também destacamos qual algoritmo encontrou a melhor solução.

Inst	melhor alg	k	k opt	min comp.	comp. opt
X-n491-k59	59	59	144986	66510	A
X-n502-k39	42	39	96991	69230	A
X-n513-k21	21	21	61091	24201	C
X-n524-k153	160	153	252534	154594	A
X-n536-k96	98	96	200034	94988	A
X-n548-k50	55	50	143275	86701	A
X-n561-k42	42	42	125252	42722	A
X-n573-k30	30	30	71365	50719	C
X-n586-k159	171	159	358838	190423	A
X-n599-k92	95	92	230881	108490	A
X-n613-k62	62	62	170670	59556	A
X-n627-k43	44	43	132331	62210	C
X-n641-k35	35	35	120534	63737	C
X-n655-k131	164	131	202778	106780	A
X-n670-k130	137	130	336762	146477	B
X-n685-k75	75	75	194121	68261	A
X-n701-k44	45	44	151035	81934	C
X-n716-k35	35	35	80182	43414	C
X-n733-k159	168	159	381616	136250	B
X-n749-k98	99	98	174922	77365	A
X-n766-k71	72	71	211319	114525	A
X-n783-k48	48	48	165241	72445	C
X-n801-k40	43	40	144666	73331	A
X-n819-k171	177	171	363760	158267	A
X-n837-k142	148	142	361034	193813	A
X-n856-k95	107	95	203672	89007	A
X-n876-k59	59	59	193000	99331	A
X-n895-k37	38	37	133319	53946	C
X-n916-k207	221	207	657966	329247	B
X-n936-k151	157	151	290600	132926	A
X-n957-k87	96	87	189370	85482	A
X-n979-k58	58	58	196197	119008	A
X-n1001-k43	43	43	105525	72404	C

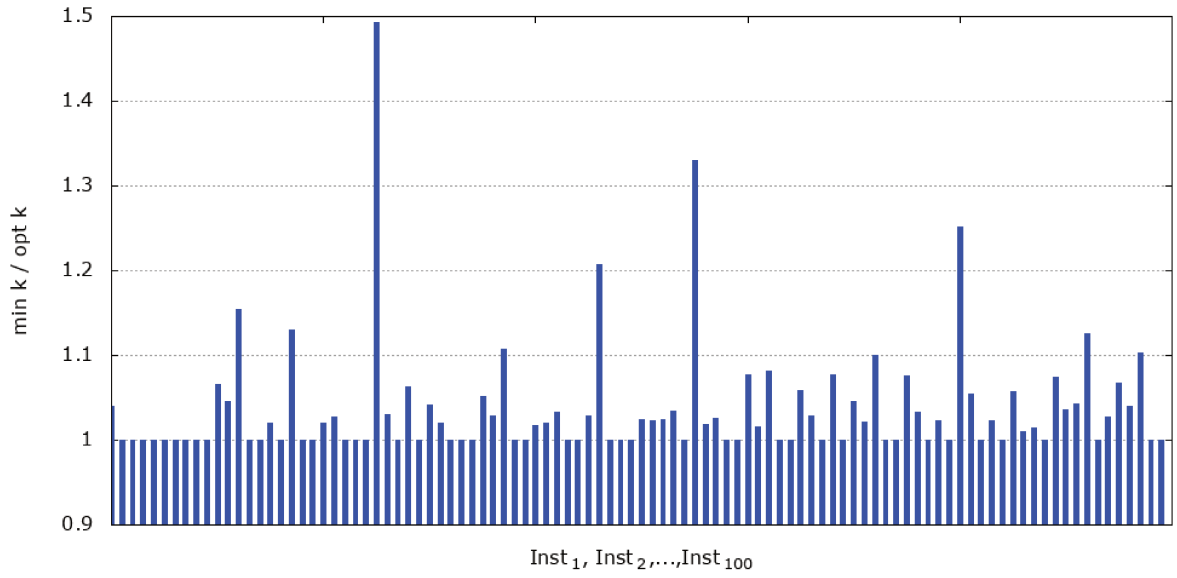


Figura 5.14: Comparação do melhor k obtido pelo AG em relação ao k ótimo para cada uma das 100 instâncias do conjunto do X. Da esquerda para direita, cada barra corresponde a uma das instâncias, seguindo ordem crescente de n.

Capítulo 6

Conclusões

Este trabalho propõe algoritmos baseados em AG e busca local para resolver o VRP. Trabalhamos com duas variantes do problema, o PostVRP e o CVRP. No primeiro, o comprimento da rota é limitada enquanto no segundo, a capacidade do veículo é limitada. Além disso, o PostVRP trabalha com três objetivos, número de veículos, comprimento da rota e variância entre os comprimentos das rotas, enquanto o CVRP trabalha minimizando o comprimento de rota e, de maneira secundária, o número de veículos.

Optamos por desenvolver algoritmos genéticos para resolver o problema. O algoritmo proposto é um AG canônico com busca local. O cromossomos escolhidos não possuem divisão de rotas. No momento do *crossover*, é executado um algoritmo construtivo que faz a delimitação das rotas. Na opção LS *Enabled*, é executado um algoritmo de melhoria incremental da solução.

Este trabalho comparou o desempenho de um algoritmo genético usando dois operadores diferentes de *crossover*, assim como a combinação do AG com busca local 2-opt. Na maioria dos casos o operador BCR obteve resultados melhores e o uso da busca local proporcionou uma convergência mais rápida aos algoritmos. Ainda assim, o operador OX em conjunto ao 2-opt se

mostrou uma opção viável, não só por obter resultados próximos à melhor configuração, mas também por ter um tempo de execução menor.

Em relação ao conjunto de instâncias X, os algoritmos propostos obtiveram o número ótimo de veículos em 46% dos casos e ficaram acima de 10% piores em apenas 9%, com destaque à configuração BCR LS *Enabled*, que obteve o melhor desempenho em 57% dos casos. Para que um desempenho similar possa ser alcançado em relação ao comprimento, a função de *fitness* do AG deve ser modificada.

Em relação ao conjunto de instâncias PostVRP, mais experimentos precisam ser conduzidos, especialmente para os casos maiores a partir de 5.000 pontos de entrega. Tais instâncias parecem necessitar de mais de 100 gerações. O algoritmo não foi testado para a maioria dos casos com mais de 10.000 de entrega. Além disso, é importante que futuros experimentos trabalhem com todas as três funções objetivo do problema.

Referências bibliográficas

ALTINKEMER, K.; GAVISH, B. Parallel Savings Based Heuristics for the Delivery Problem. **Operations Research**, INFORMS, v. 39, p. 456–469, 1991. ISSN 0030364X, 15265463. Disponível em: <http://www.jstor.org/stable/171399>.

APPLEGATE, D. L.; BIXBY, R. E.; CHVATAL, V.; COOK, W. J. **The traveling salesman problem: a computational study**. [S.l.]: Princeton university press, 2006.

APPLEGATE, D. L.; COOK, W. J.; ROHE, A. Chained Lin-Kernighan for large traveling salesman problems. **INFORMS Journal on Computing**, INFORMS, v. 15, p. 82–92, 2003.

ASSAF, R.; SALEH, Y. Vehicle-routing optimization for municipal solid waste collection using genetic algorithm: the case of southern Nablus city. **Civil and Environmental Engineering Reports**, De Gruyter Open, v. 26, p. 43–57, 2017.

AUGERAT, P. H.; BELENGUER, J. M.; BENAVENT, E.; CORBERÁN, A.; NADDEF, D.; RINALDI, G. **Computational results with a branch and cut code for the capacitated vehicle routing problem**. [S.l.]: IMAG, 1995.

BACK, T. **Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms**. [S.l.]: Oxford university press, 1996.

BEASLEY, J. E. Route first—cluster second methods for vehicle routing. **Omega**, Elsevier, v. 11, p. 403–408, 1983.

BORTHEN, T.; LOENNECHEN, H.; WANG, X.; FAGERHOLT, K.; VIDAL, T. A genetic search-based heuristic for a fleet size and periodic routing problem with application to offshore supply planning. **EURO Journal on Transportation and Logistics**, v. 7, 2018.

BREEDAM, A. V. **An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints**. 1994. Tese (Doutorado) – University of Antwerp, Belgium.

CAO, W.; YANG, W. A Survey of Vehicle Routing Problem. **MATEC Web of Conferences**, v. 100, 2017. DOI: 10.1051/matecconf/201710001006.

CESCHIA, S.; SCHAERF, A.; STÜTZLE, T. Local search techniques for a routing-packing problem. **Computers & industrial engineering**, Elsevier, v. 66, p. 1138–1149, 2013.

CHRISTOFIDES, N. The vehicle routing problem. **Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche Opérationnelle**, v. 10, p. 55–70, 1976. Disponível em: <http://www.numdam.org/item?id=RO_1976__10_1_55_0>.

— — [s.l.]: Wiley & Sons, 1979.

CHRISTOFIDES, N.; EILON, S. An algorithm for the vehicle-dispatching problem. **Journal of the Operational Research Society**, JSTOR, p. 309–318, 1969. Disponível em: <https://doi.org/10.1057/jors.1969.75>.

CLARKE, G.; WRIGHT, J. W. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. **Operations Research**, v. 12, p. 568–581, 1964. DOI: 10.1287/opre.12.4.568. eprint: https://doi.org/10.1287/opre.12.4.568. Disponível em: <https://doi.org/10.1287/opre.12.4.568>.

COOK, W. J. **In pursuit of the traveling salesman: mathematics at the limits of computation**. [S.l.]: Princeton University Press, 2011.

CRESCENZI, P.; KANN, V.; HALLDÓRSSON, M.; KARPINSKI, M.; WOEGINGER, G. A compendium of NP optimization problems. **URL: http://www. nada. kth. se/˜ viggo/problemlist/compendium. html**, 1997.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management science**, INFORMS, v. 6, p. 80–91, 1959. Disponível em: <https://doi.org/10.1287/mnsc.6.1.80>.

DIJKSTRA, E. W. A Note on Two Problems in Connexion with Graphs. **Numer. Math.**, Springer-Verlag New York, Inc., v. 1, p. 269–271, 1959. ISSN 0029-599X. DOI: 10.1007/BF01386390. Disponível em: <http://dx.doi.org/10.1007/BF01386390>.

FARAHANI, R.; REZAPOUR, S.; KARDAR, L. **Logistics operations and management: concepts and models**. [S.l.]: Elsevier, 2011. cap. 8. ISBN 9780123852038.

FISHER, M. L. Optimal solution of vehicle routing problems using minimum k-trees. **Operations research**, INFORMS, v. 42, p. 626–642, 1994. Disponível em: <https://doi.org/10.1287/opre.42.4.626>.

FISHER, M. L.; JAIKUMAR, R. A generalized assignment heuristic for vehicle routing. **Networks**, Wiley Subscription Services, Inc., A Wiley Company, v. 11, p. 109–124, 1981. ISSN 1097-0037. DOI: 10.1002/net.3230110205. Disponível em: <http://dx.doi.org/10.1002/net.3230110205>.

FOSTER, B. A.; RYAN, D. M. An Integer Programming Approach to the Vehicle Scheduling Problem. **Operational Research Quarterly (1970-1977)**, Palgrave Macmillan Journals, v. 27, p. 367–384, 1976. ISSN 00303623. Disponível em: <http://www.jstor.org/stable/3009018>.

FRANCIS, P. M.; SMILOWITZ, K. R.; TZUR, M. The period vehicle routing problem and its extensions. In: THE vehicle routing problem: latest advances and new challenges. [S.l.]: Springer, 2008. p. 73–102.

FUKASAWA, R.; LONGO, H.; LYSGAARD, J.; ARAGÃO, M. P.; REIS, M.; UCHOA, E.; WERNECK, R. F. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. **Mathematical programming**, Springer, v. 106, p. 491–511, 2006. Disponível em: <https://doi.org/10.1007/s10107-005-0644-x>.

GENDREAU, M.; IORI, M.; LAPORTE, G.; MARTELLO, S. A Tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. **Networks**, Wiley Subscription Services, Inc., A Wiley Company, v. 51, p. 4–18, 2008. ISSN 1097-0037. DOI: 10.1002/net.20192. Disponível em: <http://dx.doi.org/10.1002/net.20192>.

GILLETT, B. E.; MILLER, L. R. A Heuristic Algorithm for the Vehicle-Dispatch Problem. **Operations Research**, v. 22, p. 340–349, 1974. DOI: 10.1287/opre.22.2.340. eprint: https://doi.org/10.1287/opre.22.2.340. Disponível em: <https://doi.org/10.1287/opre.22.2.340>.

GOLDEN, B. L.; WASIL, E. A.; KELLY, J. P.; CHAO, I. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: FLEET management and logistics. [S.l.]: Springer, 1998. p. 33–56. Disponível em: <https://doi.org/10.1007/978-1-4615-5755-5_2>.

HOKAMA, P. H. D. B. et al. Algoritmos para problemas com restrições de empacotamento. [sn], 2016.

HOROWITZ, E.; SAHNI, S. **Fundamentals of Computer Algorithms**. [S.l.]: Computer Science Press, 1978.

JOHNSON, D. S.; MCGEOCH, L. A. The traveling salesman problem: A case study in local optimization. **Local search in combinatorial optimization**, Chichester, UK, v. 1, p. 215–310, 1997.

KALLEHAUGE, B.; LARSEN, J.; MADSEN, O. B. G.; SOLOMON, M. M. **Vehicle routing problem with time windows**. [S.l.]: Springer, 2005. Disponível em: <https://doi.org/10.1007/0-387-25486-2_3>.

KARAKATIČ, S.; PODGORELEC, V. A survey of genetic algorithms for solving multi depot vehicle routing problem. **Applied Soft Computing**, Elsevier, v. 27, p. 519–532, 2015. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2014.11.005. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494614005572>.

KIM, G.; ONG, Y. S.; HENG, C. K.; TAN, P. S.; ZHANG, N. A. City Vehicle Routing Problem (City VRP): A Review. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 16, p. 1654–1666, 2015. ISSN 1524-9050. DOI: 10.1109/TITS.2015.2395536.

LAWLER, E. L.; LENSTRA, J. K.; KAN, A. H. G. R.; SHMOYS, D. B. **The traveling salesman problem: a guided tour of combinatorial optimization**. [S.l.]: Wiley New York, 1985. v. 3.

LI, F.; GOLDEN, B.; WASIL, E. Very large-scale vehicle routing: new test problems, algorithms, and results. **Computers & Operations Research**, Elsevier, v. 32, p. 1165–1179, 2005. Disponível em: <https://doi.org/10.1016/j.cor.2003.10.002>.

LIN, S. Computer solutions of the traveling salesman problem. **The Bell System Technical Journal**, Wiley Online Library, v. 44, p. 2245–2269, 1965. ISSN 0005-8580. DOI: 10.1002/j.1538-7305.1965.tb04146.x.

LIN, S.; KERNIGHAN, B. W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. **Operations Research**, INFORMS, v. 21, p. 498–516, 1973. ISSN 0030364X, 15265463. Disponível em: <http://www.jstor.org/stable/169020>.

MÄNNEL, D.; BORTFELDT, A. A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints. **European Journal of Operational Research**, v. 254, p. 840–858, 2016. Disponível em: <https://ideas.repec.org/a/eee/ejores/v254y2016i3p840-858.html>.

MEIRA, L. A. A.; MARTINS, P. S.; MENZORI, M.; ZENI, G. A. Multi-Objective Vehicle Routing Problem Applied to Large Scale Post Office Deliveries. **ArXiv e-prints**, 2018. arXiv: 1801.00712 [cs.AI].

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, Elsevier, v. 24, p. 1097–1100, 1997.

MOLE, R. H.; JAMESON, S. R. A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion. **Operational Research Quarterly (1970-1977)**, Palgrave Macmillan Journals, v. 27, p. 503–511, 1976. ISSN 00303623. Disponível em: <http://www.jstor.org/stable/3008819>.

OLIVER, I. M.; SMITH, D. J.; HOLLAND, J. R. C. A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In: PROCEEDINGS of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application. [S.l.]: L. Erlbaum Associates Inc., 1987. p. 224–230. ISBN 0-8058-0158-8. Disponível em: <http://dl.acm.org/citation.cfm?id=42512.42542>.

OMBUKI, B.; ROSS, B. J.; HANSHAR, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. **Applied Intelligence**, Springer, v. 24, p. 17–30, 2006.

OSMAN, I. H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. **Annals of operations research**, Springer, v. 41, p. 421–451, 1993.

PECIN, D. G. **Exact algorithms for the capacitated vehicle routing problem**. 2014. Tese (Doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

PEREIRA, F. B.; TAVARES, J. **Bio-inspired algorithms for the vehicle routing problem**. [S.l.]: Springer, 2008. v. 161.

RENAUD, J.; LAPORTE, G.; BOCTOR, F. F. A tabu search heuristic for the multi-depot vehicle routing problem. **Computers & Operations Research**, Elsevier, v. 23, p. 229–235, 1996.

SOLOMON, M. M. Algorithms for the vehicle routing and scheduling problems with time window constraints. **Operations research**, Informs, v. 35, p. 254–265, 1987. Disponível em: <https://doi.org/10.1287/opre.35.2.254>.

THOMPSON, P. M.; PSARAFTIS, H. N. Cyclic Transfer Algorithm for Multivehicle Routing and Scheduling Problems. **Operations Research**, INFORMS, v. 41, p. 935–946, 1993. ISSN 0030-364X. DOI: 10.1287/opre.41.5.935. Disponível em: <http://dx.doi.org/10.1287/opre.41.5.935>.

TOTH, P.; VIGO, D. An Overview of Vehicle Routing Problems. In: THE Vehicle Routing Problem. [S.l.: s.n.], 2002. cap. 1, p. 1–26. DOI: 10.1137/1.9780898718515.ch1. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718515.ch1. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.ch1>.

— — in: THE Vehicle Routing Problem. [S.l.: s.n.], 2002. cap. 5, p. 109–128. DOI: 10.1137/1.9780898718515.ch5. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718515.ch5. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.ch5>.

— **The vehicle routing problem**. [S.l.]: Siam, 2001.

UCHOA, E.; PECIN, D.; PESSOA, A.; POGGI, M.; SUBRAMANIAN, A.; VIDAL, T. **CVRP Library Site. Última checagem 27/10/2018.** [S.l.: s.n.], 2015. http://www.galgos.inf.puc-rio.br/vrp/.

UCHOA, E.; PECIN, D.; PESSOA, A.; POGGI, M.; VIDAL, T.; SUBRAMANIAN, A. New benchmark instances for the Capacitated Vehicle Routing Problem. **European Journal of Operational Research**, v. 257, p. 845–858, 2017. ISSN 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2016.08.012. Disponível em: <https://doi.org/10.1016/j.ejor.2016.08.012>.

VIDAL, T.; CRAINIC, T. G.; GENDREAU, M.; LAHRICHI, N.; REI, W. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. **Operations Research**, INFORMS, v. 60, p. 611–624, 2012.

ZACHARIADIS, E. E.; TARANTILIS, C. D.; KIRANOUDIS, C. T. A Guided Tabu Search for the Vehicle Routing Problem with two-dimensional loading constraints. **European Journal of Operational Research**, v. 195, p. 729–743, 2009. ISSN 0377-2217. DOI: http://dx.doi.org/10.1016/j.ejor.2007.05.058. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221707010983>.

ZENI, G. A.; MENZORI, M.; MARTINS, P. S.; MEIRA, L. A. A. VRPBench: A Vehicle Routing Benchmark Tool. **ArXiv e-prints**, 2016. https://arxiv.org/abs/1610.05402. arXiv: 1610.05402 [cs.AI].