UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS–GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEEL

DISSERTAÇÃO DE MESTRADO

# TESTE234321

NOME DO AUTOR SOBRENOME

JOINVILLE, 2020

NOME DO AUTOR SOBRENOME

# TESTE234321

Dissertação apresentada ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Nome do orientador do Sobrenome
Coorientador: Nome do coorientador Sobrenome

**Joinville**
**2020**

Sobrenome, Nome do Autor
teste234321 / Nome do Autor Sobrenome. - Joinville, 2020.
67 p. : il. ; 30 cm.

Orientador: Nome do orientador do Sobrenome
Coorientador: Nome do coorientador Sobrenome
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2020.

1. Palavra-chave. 2. Palavra-chave. 3. Palavra-chave. 4. Palavra-chave. 5. Palavra-chave. I. do Sobrenome, Nome do orientador . II. Sobrenome, Nome do coorientador . III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica. IV. Título

NOME DO AUTOR SOBRENOME

**TESTE234321**

Dissertação apresentada ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

**Banca Examinadora:**

Orientador:

---

**Prof. Nome do orientador do Sobrenome, Dr.**
Univ. XXX

Coorientador:

---

**Prof. Nome do coorientador Sobrenome, Dr.**
Univ. XXX

Membros:

---

**Prof. Professor, Dr.**
Univ. XXX

---

**Prof. Professor, Dr.**
Univ. XXX

---

**Prof. Professor, Dr.**
Univ. XXX

Joinville, 01 de maio de 2020

*Este trabalho é dedicado às crianças adultas que,*
*quando pequenas, sonharam em se tornar cientistas.*

## AGRADECIMENTOS

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz[1] e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com LaTeX fosse possível.

Agradecimentos especiais são direcionados ao Centro de Pesquisa em Arquitetura da Informação[2] da Universidade de Brasília (CPAI), ao grupo de usuários *latex-br*[3] e aos novos voluntários do grupo *abnTeX2*[4] que contribuíram e que ainda contribuirão para a evolução do abnTeX2.

[1] Os nomes dos integrantes do primeiro projeto abnTeX foram extraídos de <http://codigolivre.org.br/projects/abntex/>

[2] <http://www.cpai.unb.br/>

[3] <http://groups.google.com/group/latex-br>

[4] <http://groups.google.com/group/abntex2> e <http://www.abntex.net.br/>

*"Não vos amoldeis às estruturas deste mundo,*
*mas transformai-vos pela renovação da mente,*
*a fim de distinguir qual é a vontade de Deus:*
*o que é bom, o que Lhe é agradável, o que é perfeito.*
*(Bíblia Sagrada, Romanos 12, 2)*

**RESUMO**

Segundo a ABNT (2003, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . . ) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

**Palavras-chave**: latex. abntex. editoração de texto.

**ABSTRACT**

This is the english abstract.

**Keywords**: latex. abntex. text editoration.

## LISTA DE ILUSTRAÇÕES

**SUMÁRIO**

# 1 CONCEITOS PRELIMINARES

## 1.1 ALGORITMOS GENÉTICOS

Na década de 1960 John Holland inventou os Algoritmos Genéticos (AG) e os desenvolveu nos anos de 1960 e 1970 juntamente com seus alunos e colegas da Universidade de Michigan. E foi em 1975 através do lançamento de seu livro *"Adaptation in Natural and Artificial Systems"* , Que (HOLLAND, 1975) apresentou os AGs à comunidade . O objetivo original de Holland não era projetar algoritmos para resolver problemas específicos, mas sim estudar o fenômeno da evolução como ocorre na natureza, e desenvolver maneiras para que os mecanismos de adaptação natural pudessem ser importados para os sistemas computacionais.

Segundo (LINDEN, 2012), os algoritmos genéticos desenvolvidos inicialmente por Holland eram simples, mas conseguiam resultados satisfatórios para problemas considerados difíceis naquela época. Foi a partir de 1980 que os AGs começaram a evoluir, através da introdução de novas inovações e mecanismos cada vez mais elaborados, que tinham a intenção e necessidade de resolver, mesmo que aproximadamente, problemas práticos.

Os AGs utilizam muitas terminologias análogas às biológicas, porém suas entidades são muito mais simples do que as reais da biologia (GOLDBERG; ROTHLAUF, 2002).

- Cromossomo:é a estrutura de dados que codifica uma solução para um problema, podendo ser uma cadeia de bits, ou um vetor com as variáveis de decisão do problema;
- Indivíduo: é um membro da população, formado pelo cromossomo e sua aptidão;
- População: é o conjunto de indivíduos de uma mesma espécie, e representa um conjunto de pontos candidatos a solução do problema;
- Gene:para a biologia é a entidade de hereditariedade que é transmitida pelo cromossomo e que contém as características do organismo. Analogamente para os AGs é o parâmetro codificado no cromossomo;
- Alelo: para a biologia representa uma das características alternativas de um gene. Para os AGs, representa os valores que um gene pode assumir;
- Genótipo:para a biologia representa a composição genética, ou seja, o conjunto de genes de um indivíduo. Assim, nos AGs são representados pelas informações contidas no cromossomo;
- Fenótipo:é o cromossomo decodificado, ou seja, é o organismo que pode ser construído a partir das informações do genótipo.Segundo (LINDEN, 2012), todos os

AGs desenvolvidos para um determinado problema devem ter os seguintes elementos em comum:

- 1. uma representação, em termo de cromossomo, para as soluções candidatas a resolução do problema;
- 2. uma maneira de se criar a população inicial;
- 3. uma função de avaliação, que permite ordenar ou classificar os cromossomos de acordo com a função objetivo;
- 4. operadores genéticos, que permitam alterar a composição dos novos cromossomos gerados pelos pais, durante a reprodução;
- 5. e valores para os parâmetros que os AGs usam: tamanho da população, taxa de crossover e taxa de mutação.

Em geral os AG também devem seguir um conjunto de passos, indicados na Figura adaptado de (KONDAGESKI, 2008), onde cada ciclo completo, desde a avaliação de aptidão dos indivíduos, até a aplicação do operador de mutação, é considerado uma nova população. No decorrer do capítulo descreveremos com mais detalhes cada passo.

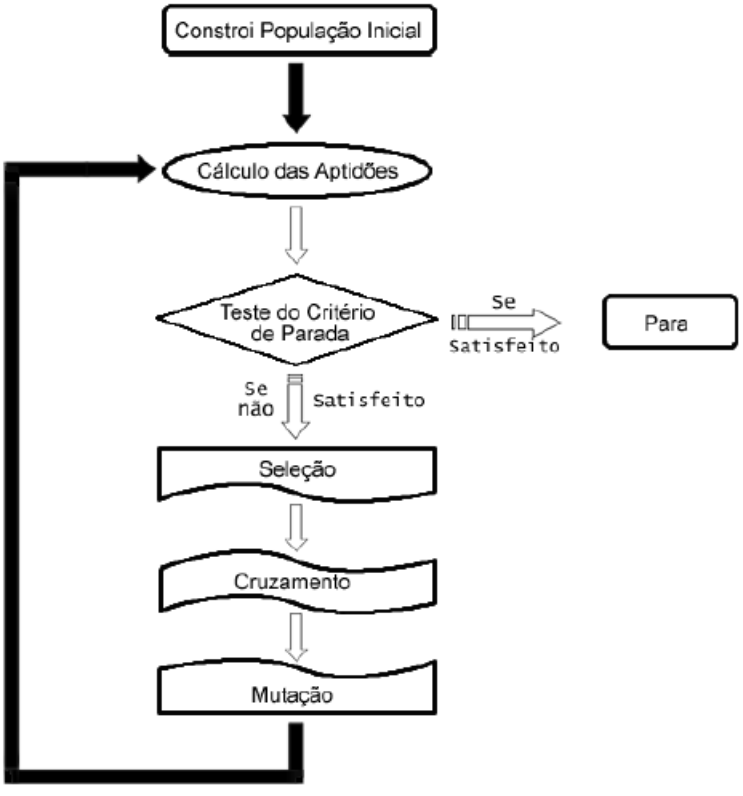


Figura 1 – Passos da execução de um AG

### 1.1.1 Representação e Codificação

Conforme vimos anteriormente todos os AGs devem possuir uma representação para as soluções candidatas a resolução do problema, chamada de cromossomo. Sua estrutura na maioria das vezes é um vetor, porém outras estruturas também podem ser utilizadas como árvores e matrizes. As codificações, ou representações mais utilizadas são a binária e a real.

Segundo (LINDEN, 2012), os primeiros cromossomos desenvolvidos por Holland em seus AGs utilizavam a codificação binária para representar os problemas. Apesar de esta codificação ser de fácil utilização e compreensão, em alguns casos obter a representação binária a partir do problema não é um processo muito óbvio, porém para um grande número de problemas de otimização esta representação é um processo natural. Devido a sua ineficiência em alguns problemas práticos com aplicações industriais, outras codificações surgiram como, por exemplo,a codificação com caracteres ou números reais.

#### *1.1.1.1 Representação Binária*

Segundo (MOGNON, 2004) o código binário, que utiliza os símbolos 0 e 1 para representar as variáveis, é muito explorado para a codificação dos cromossomos. Na codificação binária os cromossomos são compostos por strings de zeros e uns, e os parâmetros (genes) são representados por conjuntos de bits.

Porém, para (CATARINA A. S.; BACH, 2003), quando a codificação binária é utilizada em problemas com variáveis contínuas e que se espera uma boa precisão na resolução, os cromossomos se tornam longos. Consequentemente este aumento dos cromossomos resultará também em um aumento de tempo para seu processamento.

#### *1.1.1.2 Representação Real*

Uma representação alternativa em problemas de optimização com variáveis contínuas de valor real é a representação cromossomo de ponto flutuante. Com esta representação, não há necessidade de um mecanismo de codificação explícita. Cada membro de cada população do algoritmo genético é um vetor de ponto flutuante. Os operadores genéticos (mutação e *crossover*) neste caso não lidar com cadeias de bits e são definidos de uma maneira diferente. Por exemplo, a operação de mutação não altera aleatoriamente um pouco, mas aleatoriamente escolhe um número de ponto flutuante dentro de um determinado intervalo.

### 1.1.2 Criação da População Inicial

A inicialização básica de um algoritmo genético clássico se resume à síntese de uma população inicial, sobre a qual serão aplicadas as ações dos passos subsequentes do processo. Tipicamente se faz uso de funções aleatórias para gerar os indivíduos, sendo este

um recurso simples que visa a fornecer maior diversidade, fundamental para garantir uma boa abrangência do espaço de pesquisa (GOLDBERG; ROTHLAUF, 2002).

#### *1.1.2.1 inicialização de forma aleatória*

como o próprio nome já sugere, a população é gerada de forma aleatória, sem seguir nenhum padrão ou algoritmo. Pode ser considerado um método bom para testar o funcionamento de um AG, já que as soluções obtidas são puramente consequência da evolução dessa população aleatória, e não possuem interferência das características dos métodos utilizados para gerar a população inicial;

#### *1.1.2.2 inicialização heurística*

neste caso a população é gerada através de alguns métodos mais diretos como, por exemplo, o algoritmo guloso, inicialização randômica ponderada, ou mesmo uma inicialização gerada por um especialista no problema, sendo está forma muito utilizada especialmente em algumas aplicações industriais.

### 1.1.3 Função de Avaliação

Neste componente, todos os indivíduos da população sofrem um processo de avaliação, que visa a atribuição de um valor de adaptação para a solução do problema em estudo. Em conjunto com a escolha da representação, este é o ponto do algoritmo mais dependente do problema em si, pois é necessário que o AG seja capaz de responder sobre quão boa uma resposta é para o problema proposto por (GOLDBERG; ROTHLAUF, 2002).

Várias formas de avaliação são utilizadas: em casos de otimização de funções matemáticas, o próprio valor de retorno destas funções é aplicado ao indivíduo, e em problemas com muitas restrições, funções baseadas em penalidades são mais comuns. A função de avaliação é também chamada de função objetivo.

### 1.1.4 Operadores Genéticos

Após a realização da avaliação dos indivíduos e a não satisfação do critério de parada, inicia-se o processo de criação de uma nova geração, com a intenção de melhorar a aptidão dos indivíduos. Este processo de criação está fundamentado na aplicação dos operadores básicos de seleção, crossover e mutação, porém outros operadores podem ser utilizados e também variações destes considerados básicos. Nesta seção explicaremos melhor cada um dos operadores e seus principais métodos.

### 1.1.5 Operador de Seleção

Para que o processo de criação de uma nova geração inicie é necessário a seleção dos indivíduos que passarão para a nova geração, ou os chamados pais que irão gerar, através do *crossover* ,a nova geração. O operador responsável por essa classificação é o operador de seleção.

O operador de seleção assemelha-se ao processo de seleção natural, onde os indivíduos mais aptos, no caso de acordo com a função fitness , possuem uma probabilidade maior de serem selecionados. Contudo, a seleção não deve se basear apenas nos melhores indivíduos, pois estes podem não estar perto da solução ótima global, devendo também ser considerado os indivíduos com aptidão mais baixa, dando-lhes alguma chance de participar da reprodução (MOGNON, 2004).

### 1.1.6 Método da Roleta

É um dos mais utilizados para a seleção dos indivíduos. Neste método, cada indivíduo recebe uma porção da roleta de acordo com sua aptidão. Assim, consequentemente, os indivíduos com uma aptidão maior receberão também uma porção maior da roleta, o que aumentará sua probabilidade de ser selecionado. Após as atribuições, a roleta é girada e a porção sorteada corresponderá ao indivíduo selecionado, podendo ser girada quantas vezes forem necessárias de acordo com o número de indivíduos que se deseja para a próxima fase de cruzamento.

Para (MOGNON, 2004), este método tem a vantagem de todos os indivíduos terem a chance de serem selecionados. Entretanto, pode sofrer o efeito de dominância se algum indivíduo possuir uma aptidão alta em relação à média dos demais.

### 1.1.7 Método de Torneio

O primeiro passo desse método é a escolha de um subconjunto de indivíduos, que podem ser escolhidos aleatoriamente ou através de outro método, como por exemplo, o método da roleta. Esse subconjunto participará então do torneio, que consiste na competição dos indivíduos entre si considerando o seu valor de aptidão, ou seja, o vencedor será o indivíduo cujo valor de aptidão seja maior. Por fim, todos os membros do subconjunto são colocados novamente na população e o processo se repete, até que o número de indivíduos selecionados seja igual ao dos desejados (MOGNON, 2004).

#### *1.1.7.1 Operador de Crossover*

Após a seleção dos pais, um ou mais pares de indivíduos, o próximo passo para a geração dos novos indivíduos, os filhos, é a reprodução desses pares de indivíduos. O *crossover* é o operador responsável pela permutação do material genético dos pais durante

o processo reprodutivo, permitindo assim que as próximas gerações, os filhos, herdem essas características. Este operador é aplicado seguindo-se uma taxa de crossover. O operador de *crossover* pode ser implementado de diversas maneiras, dentre elas as mais conhecidas são: *crossover* de um-ponto, *crossover* multiponto e *crossover* uniforme ou pontos aleatórios

#### *1.1.7.2 Operador de Mutação*

Este operador é necessário para que na nova geração possam ser inseridos novos materiais genéticos, ou seja, diversificar o material genético, como ocorre na natureza através de mutações ou anormalidades dos indivíduos. Por isso, este operador é aplicado levando-se em consideração a taxa de mutação , que geralmente é bem pequena. É muito simples de ser implementado: cada gene é analisado, através da probabilidade de mutação, para saber se aquela posição irá alterar seu valor ou não (MOGNON, 2004).

### 1.1.8 Parâmetros Genéticos

Para que os algoritmos genéticos possam ser simulados se faz necessário a configuração de alguns parâmetros: tamanho da população, taxa de *crossover* e taxa de mutação. Segundo (CATARINA A. S.; BACH, 2003), são esses parâmetros que afetam o desempenho dos AGs, desde um aumento do tempo de convergência, até uma convergência prematura ou mesmo a não-convergência para uma solução aceitável.

#### *1.1.8.1 Tamanho da População*

Este parâmetro determina o número de indivíduos que compõem cada população. Se a população for pequena, consequentemente a cobertura do espaço de busca do problema também será, o que poderá causar uma queda no desempenho do AGs. Assim, uma população grande resultará em uma cobertura maior do espaço de busca do problema, representando realmente o domínio do problema. Contudo, para se trabalhar com esta população maior, são necessários maiores recursos computacionais, ou/e um tempo maior de processamento do algoritmo (CATARINA A. S.; BACH, 2003). Logo, este parâmetro deve ser ajustado de forma equilibrada, considerando os efeitos citados.

#### *1.1.8.2 Taxa de Crossover*

Representa a probabilidade com que ocorrerá o crossover dos indivíduos selecionados da população. Se a taxa for baixa, o algoritmo poderá se tornar muito lento, devido à inexistência de novos indivíduos. Quanto maior for esta taxa, mais rapidamente novas estruturas serão inseridas na população. Porém, se for muito alta poderá gerar efeitos indesejados, como a substituição da maior parte da população, o que poderá resultar na perda dos indivíduos de alta aptidão (CATARINA A. S.; BACH, 2003).

#### *1.1.8.3 Taxa de Mutação*

Indica a probabilidade de ocorrer a mutação na população. Segundo (CATARINA A. S.; BACH, 2003), se a taxa for baixa, ela previne que a solução fique estagnada em um valor, ou seja, a convergência prematura, e possibilita que se possa chegar em qualquer ponto do espaço de busca do problema. Se for uma taxa muito alta, tornará a busca essencialmente aleatória.

# 2 PROBLEMA ROTEAMENTO VEÍCULOS COM JANELAS DE TEMPO

## 2.1 PROBLEMA ROTEAMENTO VEÍCULOS COM JANELAS DE TEMPO

Um dos problemas mais famosos de otimização combinatória é o chamado Problema do Caixeiro Viajante, que consiste em determinar o circuito mais curto para se percorrer um dado número de pontos (chamados de nós) e retornar à origem, passando apenas uma vez por cada um deles (LIEBERMAN, 2010). Entretanto, este problema não reflete a realidade da maior parte das organizações, já que estas contam com uma série de veículos, que experimentam diversas restrições (de tempo e capacidade, por exemplo) e percorrem rotas distintas. Logo, o desafio destas empresas é determinar a melhor alocação dos veículos disponíveis, resolvendo um problema de roteamento de veículos (PRV).

O Problema de Roteamento de Veículos (PRV) é descrito como o problema de planejar a entrega ou coleção de rotas ótima. Estas rotas são compostas por veículos que devem partir de um ou vários depósitos para um determinado número de cidades ou clientes espalhados geograficamente, sujeito a um conjunto de restrições.

(LAPORTE, 1992) define o Problema Clássico de Roteamento de Veículos e mostra uma visão geral das diversas abordagens utilizadas para solucioná-lo. Estas se desdobram em algoritmos exatos, que encontram a solução ótima para o problema, e algoritmos heurísticos, que buscam uma boa solução viável, mas que não é necessariamente a solução ótima.

O PRV pode ser definido da seguinte forma: Seja um grafo onde é um conjunto de vértices representando localidades (clientes ou cidades) com o depósito localizado no vértice , e é o conjunto de arcos. Cada arco , é associado a uma matriz de distâncias não negativas. Em alguns contextos, também pode ser interpretado como o custo de viagem ou o tempo de viagem. Quando é simétrico (isto é, a distância/tempo/custo de para é o mesmo de para ), é conveniente substituir por um conjunto de arcos não direcionados. Além disso, assumimos que existem veículos disponíveis no depósito, faz sentido associar um custo fixo ao uso do veículo. Como simplificação, (LAPORTE, 1992) ignorou estes custos, e partiu-se do princípio de que todos os veículos são idênticos e têm a mesma capacidade . O PRV consiste em planejar um conjunto de rotas de menor custo do veículo, de tal forma que:

- Cada vértice em é visitado apenas uma vez e por exatamente um veículo;
- Todas as rotas se iniciam e terminam no depósito;
- As seguintes restrições devem ser respeitadas:

- Restrição de capacidade: a cada vértice é atribuído um peso não-negativo (ou demanda) e a soma dos pesos de qualquer rota do veículo não pode exceder a capacidade do veículo;
- O número de vértices em cada rota é limitado a (este é um caso especial de (a) com para todo e );
- Restrição de tempo total: o comprimento de qualquer rota não pode exceder um limite fixado , sendo este comprimento constituído pelos tempos de viagem e pelos tempos de parada em cada vértice da rota;
- Janelas de tempo: o vértice deve ser visitado dentro do intervalo de tempo e é permitido tempo de espera no vértice ;
- Precedência entre pares de vértices: o vértice pode ter de ser visitado antes do vértice .

Esta lista não é exaustiva, e uma série de outras variantes interessantes são descritas na literatura.

O Problema de Roteamento de Veículos(PRV) com Janelas de Tempo é uma extensão do PRV onde os serviços de cada cliente devem começar associados a um intervalo de tempo , chamado janela de tempo(*time window*) . As janelas de tempo podem ser rígidas ou flexíveis. em caso de rígida , um veiculo que chega no cliente muito cedo deve esperar até que o cliente esteja pronto para começar o serviço. Em geral , esperar antes do início de uma janela de tempo não incorre em custos. No caso de janelas de tempo flexíveis , cada janela de tempo pode ser violada incorrendo um custo penalização.As janelas de tempo podem ser unilaterais, por exemplo, o tempo máximo para o inicio de uma ação.

Janelas de tempo surgem naturalmente em problemas enfrentados por organizações empresariais que trabalham com horários flexíveis . Problemas específicos com janelas de tempo rígida incluem serviço de segurança e patrulha, entregas bancárias, envios postais, recolhimento de lixo e roteamento de ônibus . Entre os problemas da janela de tempo flexíveis , problemas de entrega de encomendas constituem um exemplo importante. Neste capítulo, vamos nos concentrar principalmente nas Janelas de tempo rígidas.

Na literatura existente sobre o PRVJT , o número de veículos disponíveis para servir os clientes é geralmente considerada ilimitada e a função de objectivo depende da natureza do método de solução escolhida. Para métodos exatos o objetivo é o de minimizar a distância total percorrida. Para heurísticas o principal objetivo é minimizar o número de veículos utilizados e o secundário para minimizar a distância total percorrida. Pode haver excepções a esta declaração geral.

Desde que o PRV é NP-hard , por restrição, PRVJT também é NP-hard (LAPORTE; TOTH; VIGO, 2013).Na verdade até mesmo encontrar uma solução viável para o PRVJT

para um número fixo de veículos é em si um problema NP-completo (SAVELSBERGH; SOL, 1995).Uma janela de tempo curta é uma janela que influencia a solução; ou seja, a janela é uma restrição ativa , já as janelas de tempo longas são restrições que influenciam menos nos resultados.

No VRP a geografia é geralmente o fator importante que determina a forma das rotas. Se o número de clientes é pequena, um despachante treinado pode fazer muito bem no planejamento das rotas somente olhando um mapa mostrando a localização dos clientes.No entanto, se a restrição de capacidade é obrigatória para algumas das rotas, é muito mais difícil de ignorar a situação de planejamento.Para o PRVJT quando algumas das rotas são limitadas pela capacidade e outras pelas janelas de tempo, é ainda mais difícil de planejar as rotas manualmente.A interação entre o espaço e os elementos temporais das rotas pode resultar em rotas ótimas que estão longe de a imagem clássica de rotas formadas.Se o número de clientes é aumentada para um nível realista, dizem que pelo menos 100-200, torna-se muito difícil fazer as rotas manualmente. É aqui que os métodos de solução computacionais mostram as suas vantagens(LAPORTE; TOTH; VIGO, 2013).

Os primeiros trabalhos sobre a PRVJT foram estudo de caso orientado (PULLEN; WEBB, 1967) e (KNIGHT; HOFER, 1968). Os métodos de solução com base em heurísticas foram relativamente simples.Os primeiros algoritmos exatos de Branch-and-Bound surgiram no início da década de 1980 (LAPORTE; TOTH; VIGO, 2013) e (KOLEN; KAN; TRIENEKENS, 1987).Em 1987, (SOLOMON, 1987) introduziu instâncias de benchmark envolvendo 100 clientes que foram aceitas como problemas de benchmark padrão pela maioria dos pesquisadores que trabalham no PRVJT e serviu como um catalisador para aumentar a pesquisa sobre o PRVJT.Na década seguinte, muitas heurísticas foram desenvolvidas, as buscas na sua maioria locais, mas também as primeiras metaheurísticas (busca tabu e algoritmos genéticos).Vários algoritmos exatos baseados em metodologias complexas, como relaxamento de Lagrange e de geração de colunas, também foram concebidos.

## 2.2 FORMULAÇÃO MATEMÁTICA

O PRVJT é definido no gráfico dirigido G = (V,A) em que o depósito é representado pelos dois vértices 0 e n + 1 , Referido como os vértices partida e destino, respectivamente .Seja N=$V/\{0, n+1\}$ são o conjunto de vértices do cliente.Todas as rotas de veículos viáveis correspondem caminhos elementares em G.O inverso é, no entanto, não é necessariamente verdade;ou seja, alguns caminhos elementares em G pode não representar rotas viáveis porque violam as janelas de tempo ou a capacidade do veículo.Para simplificar a notação, zero demandas e zero tempos de serviço são definidos para vértices 0 e n + 1 .Além disso, uma janela de tempo está associada com eles exemplo $(a0, b0) = (an + 1, bn + 1)$ onde a0 e b0 são o mais cedo possível saída do depósito e o último horário possível chegada no depósito,respectivamente .Supondo-se que a matriz de tempo de viagem satisfaz

a desigualdade triângulo, existem soluções viáveis apenas se $a_0 \leq min_{i \in V/\{0\})}\{b_i t_{0i}\}$ e $b_0 \geq max_{i \in V/\{0\})}\{max\{a_0 t_{0i}, a_i\} + s_i + t_{i,n+1}\}$

Note que um arco (i,j) ∈ A pode ser omitida devido a considerações temporais, se $a_i + s_i + t_{ij} > b_j$ ou limitações de capacidade $q_i + q_j$ ou por outros fatores.Finalmente, vamos falar que quando os veículos são autorizados a permanecer no depósito, especialmente no caso em que o principal objetivo consiste em minimizar o número de veículos usados, o arco (0,n +1) com $c_{0,n+1=t_{0,n+1=0}}$ deve ser adicionado ao conjunto de arcos A.

Primeiro apresentaremos uma formulação usando Programação Inteira Mista(PIM) para o PRVJT envolvendo dois tipos de variáveis: para cada arco(i,j)∈ A e cada veiculo k ∈ K há um arco variável de fluxo binário $x_{ijk}$ que é igual a 1 se o arco (i, j) é utilizada pelo veículo k, e 0 de outro modo;e, para cada vértice i ∈ V e veiculo k ∈ K, temos uma variável de tempo $T_{ik}$ especificando o início do tempo de serviço no vértice i quando servida por veículo k.

O PRVJT pode ser formulado como o seguinte modelo de fluxo de rede de multi-produto com limitações de janela de tempo e de capacidade:

$$\text{(3.1)} \quad \text{(VRPTW1) minimize} \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk}$$

$$\text{(3.2)} \quad \text{s.t.} \sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ijk} = 1 \qquad \forall\, i \in N,$$

$$\text{(3.3)} \quad \sum_{j \in \delta^+(0)} x_{0jk} = 1 \qquad \forall\, k \in K,$$

$$\text{(3.4)} \quad \sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0 \qquad \forall\, k \in K,\ j \in N,$$

$$\text{(3.5)} \quad \sum_{i \in \delta^-(n+1)} x_{i,n+1,k} = 1 \qquad \forall\, k \in K,$$

$$\text{(3.6)} \quad x_{ijk}(T_{ik} + s_i + t_{ij} - T_{jk}) \leq 0 \qquad \forall\, k \in K,\ (i,j) \in A,$$

$$\text{(3.7)} \quad a_i \leq T_{ik} \leq b_i \qquad \forall\, k \in K,\ i \in V,$$

$$\text{(3.8)} \quad \sum_{i \in N} q_i \sum_{j \in \delta^+(i)} x_{ijk} \leq Q \qquad \forall\, k \in K,$$

$$\text{(3.9)} \quad x_{ijk} \in \{0,1\} \qquad \forall\, k \in K,\ (i,j) \in A.$$

função objetivo (3.1) visa minimizar o custo total. As restrições (3.2) garantem que cada cliente é atribuído a exatamente um percurso.Em seguida, as restrições (3.3) - (3.5) definem um caminho da *source-to-sink* no G para cada k veículo.Além disso, as restrições (3.6) - (3.7) e a viabilidade do cronograma (3.8) garantia em relação à janelas de tempo e capacidade do veículo, respetivamente. Finalmente, os arcos de fluxo estão sujeitos a requisitos binários (3.9).

Modelo (3.1) - (3.9) é não linear devido as restrições de (3.6) que pode, no entanto, ser linearizada como:

$$T_{ik} + s_i + t_{ij} - T_{jk} \leq (1 - x_{ijk})M_{ij} \qquad \forall\, k \in K, (i,j) \in A, \tag{3.6a}$$

Onde $M_{ij}$ , (i, j ) ∈ A, são grandes constantes que podem ser definidas para $max\{b_i + s_i + t_{ij} - a_j, 0\}$

O relaxamento linear do modelo (3.1) - (3.5), (3.6a), (3.7) - (3.9) prevê, em geral, limites inferiores muito fracos . Este modelo tem, no entanto, uma estrutura de bloco-angular que pode ser explorada, onde cada bloco é composto das restrições (3.3) - (3.5), (3.6a), (3.7) - (3.9) para um veículo específico, k ∈ K e define um Problema do Caminho Mais Curto Elementar com restrições de recursos (ESPPRC). Aplicando o princípio de decomposição de Dantzig-Wolfe (GENDREAU; LAPORTE; POTVIN, 2001) para este modelo produz o seguinte modelo de particionamento definida uma vez que os veículos (idênticos) e suas variáveis correspondentes são agregados (GENDREAU; POTVIN, 2010). Neste modelo,$\omega$ denota o conjunto de todas as rotas possíveis, $c_r$ o custo da rota r ∈ $\omega$ and $a_{ir}$ o número de visitas ao cliente i ∈ N em rota r ∈ $\omega$ ($a_{ir}$ ∈ 0,1, quando r é uma rota fundamental. Com cada rota r ∈ $\omega$ está associado um ano variável de caminho binária que assume valor 1 se rota r é selecionado na solução e 0 caso contrário. o modelo de partição do conjunto é

$$\text{(VRPTW2) minimize} \sum_{r \in \Omega} c_r y_r \tag{3.10}$$

$$\text{s.t.} \sum_{r \in \Omega} a_{ir} y_r = 1 \qquad \forall\, i \in N, \tag{3.11}$$

$$y_r \in \{0,1\} \qquad \forall\, r \in \Omega. \tag{3.12}$$

função objetivo (3.10) procura minimizar o custo total. Definir restrições de particionamento (3.11) impõem que cada cliente ser visitada apenas uma vez por um veículo. Os requisitos binários nas variáveis caminho de fluxo são expressos por (3.12). Note-se que, como na literatura PRVJT, o modelo acima assume que o número de veículos disponível para atender os clientes é ilimitada, isto é, |k| é tão grande quanto necessário. Se este não foi o caso, a restrição de aplicar a seleção de, no máximo, uma rota disponível por veículo iria ser adicionado ao modelo.

$$\sum_{r \in \Omega} y_r \leq |K| \tag{3.13}$$

O relaxamento linear do modelo (3.10) - (3.12) produz uma melhor limites mais baixos do que a do modelo (3.1) - (3.5), (3.6a), (3.7) - (3.9). Por outro lado, o modelo de partição do conjunto contém um grande número de variáveis, por rota viável.

Vários outros modelos foram propostos para a PRVJT. Em particular, as formulações de dois índices foram usados em conjunto com algoritmos Branch-and-CUT . Apresenta-se uma tal formulação abaixo, que envolve um tipo de variáveis: para cada arco (i, j) ∈ A, existe um binário variável $x_{ij}$ que é igual a 1 se o arco (i, j) é utilizada em solução e 0 se não for usado. Denote por P o conjunto de caminhos (não necessariamente a partir da origem) em G que não respeitam as restrições de janela de tempo e por A (p) o conjunto de arcos no caminho p∈ P. Deixe r(S) o número mínimo de veículos necessários para servir cada subconjunto de clientes S de acordo com suas demandas. Este número é, em geral, substituídos por $[q(S)/Q]$ em $q(S) = \sum_{i \in S} q_i$. A formulação de duas índice corresponde à

$$\text{(3.14)} \qquad \text{(VRPTW3) minimize} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\text{(3.15)} \qquad \text{s.t.} \sum_{j\in\delta^-(i)} x_{ji} = 1 \qquad \forall\, i \in N,$$

$$\text{(3.16)} \qquad \sum_{j\in\delta^+(i)} x_{ij} = 1 \qquad \forall\, i \in N,$$

$$\text{(3.17)} \qquad \sum_{i\notin S}\sum_{j\in S} x_{ij} \geq r(S) \qquad \forall\, S \subseteq N,\ S \neq \emptyset,$$

$$\text{(3.18)} \qquad \sum_{(i,j)\in A(p)} x_{ij} \leq |A(p)| - 1 \qquad \forall\, p \in P,$$

$$\text{(3.19)} \qquad x_{ij} \in \{0, 1\} \qquad \forall\, (i, j) \in A.$$

função objetivo (3.14) minimiza o custo total para servir os clientes. Restrições (3.15) - (3.16) garantir que um veículo chega e sai de cada cliente, respectivamente. desigualdades de capacidade (3.17) garantir que a capacidade do veículo está satisfeito em todas as rotas selecionadas e também força o relaxamento linear através da imposição de um número mínimo de veículos para atender cada subconjunto de clientes S. Além disso, eles agem como restrições de eliminação sub deslocamento. desigualdades caminho inviável (3.18) proibir a seleção de caminhos que não respeitam as janelas de tempo. Finalmente, as variáveis de fluxo $x_{ij}$ estão sujeitos a requisitos de binários (3.19).

A formulação de dois índice (3.14) - (3.19) contém um número exponencial de restrições (3.17) e (3.18). Para os casos de tamanhos práticos , eles precisam ser gerados dinamicamente como no algoritmo de plano de corte. desigualdades válidas adicionais também podem ser considerados para apertar o relaxamento linear deste modelo

### 2.2.1 Conjunto de Problemas Testes de Solomon

O conjunto de problemas teste proposto por Solomon em 1987 (SOLOMON, 1987), baseado em dados de alguns problemas usados por Christofides et al. (1979) para o problema de roteamento padrão, trata-se de diferentes classes de instâncias, cada qual com características geográficas e de restrições características. Os consumidores estão distribuídos em um plano XY de dimensões 100x100.

As instâncias são divididas em 6 grupos denominados R1, R2, C1, C2, RC1 e RC2. Cada classe contém entre 8 e 12 instâncias. Os grupos R1 e R2, possuem uma disposição geográfica dos clientes de forma aleatória, já nos grupos C1 e C2, a disposição é na forma de agrupamentos, e nos grupos RC1 e RC2 são tipos mistos (parte em agrupamentos e parte aleatória). Nas classes R1, C1 e RC1, as janelas de tempo e o horizonte total são curtos, diminuindo assim o número de consumidores por rota. Já as classes R2, C2 e RC2, possuem um longo horizonte total fazendo com que as rotas tenham mais consumidores viáveis.

Cada instância tem 100 consumidores, mas pode-se considerar apenas os primeiros 25 ou 50 consumidores dependendo do caso.

No caso do uso das instâncias de Solomon, nos problemas de roteamento dinâmico, adaptações devem ocorrer para que não se tenha conhecimento de todos os consumidores no início do roteamento, o que tornaria o problema estático.

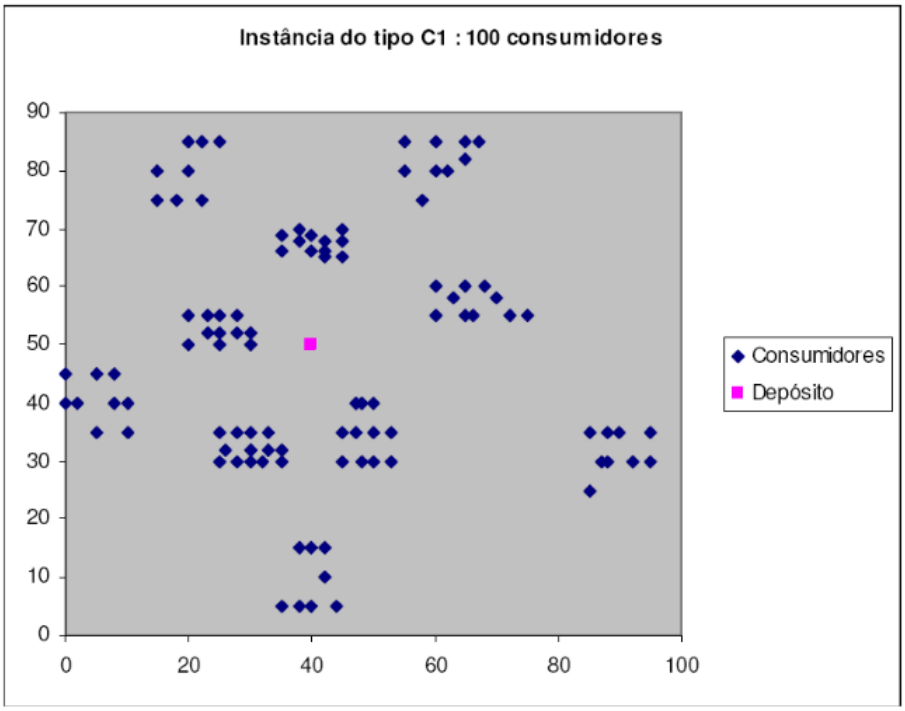


Figura 2 – Problemas classe C

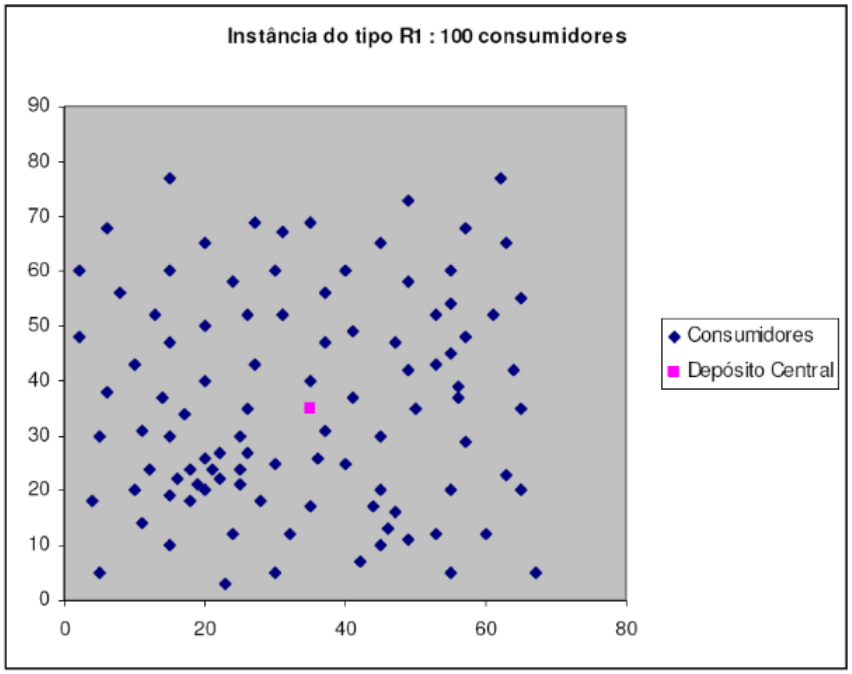


Figura 3 – Problemas classe R

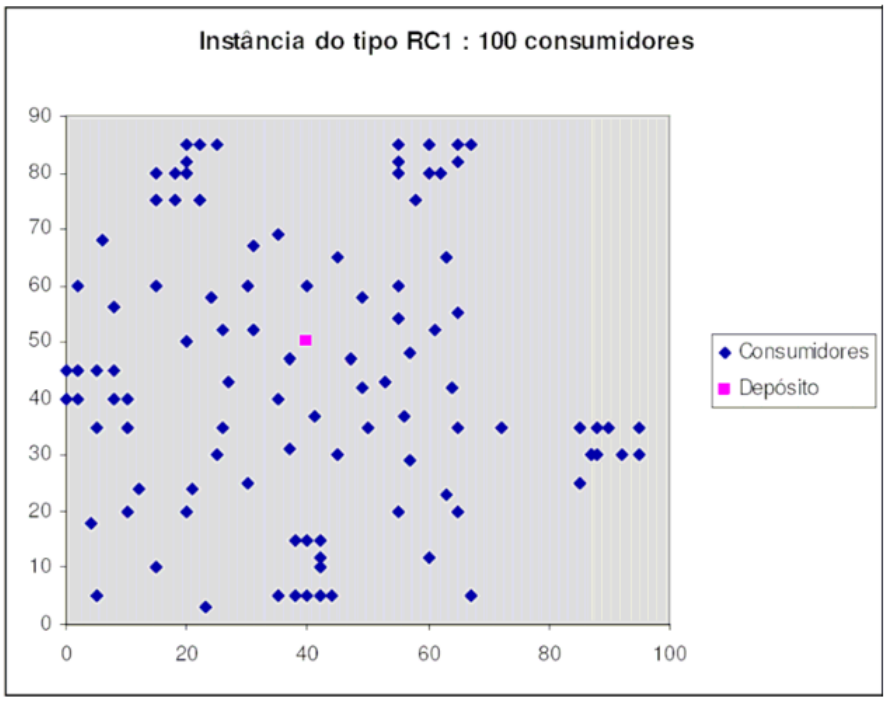


Figura 4 – Problemas classe RC

# 3 MÉTODOS DE SOLUÇÃO

## 3.1 MÉTODOS DE SOLUÇÃO EXATA

Os métodos de solução exata visam identificar uma solução ótima para uma dada instância.Investigam todas as soluções admissíveis ou de garantem que não há necessidade de investigar outras soluções, pois as soluções não analisadas não originarão em soluções melhores que as já foram encontradas.No entanto, como referido, e dada a complexidade deste tipo de problemas, há instâncias que, pela sua dimensão, sera necessário muito tempo computacional para poderem ser resolvidas. Assim, os métodos exatos são, regra geral, apenas aplicados a instâncias de pequena ou média dimensão. De seguida, apresentam-se três destes métodos.

### 3.1.1 *Branch-and-Bound*

O método *Branch-and-Bound*(BeB), proposto por (LAND A. H., 1960)consiste em dividir um dado problema em vários sub-problemas de menor dimensão e de mais fácil resolução garantindo que a resolução destes problemas mais fáceis conduz à solução do problema inicial. Estas divisões são realizadas iterativamente, tendo em conta que os sub-problemas a resolver devem ser mais fáceis do que o problema que os originou. Resolvido um sub-problema, é analisada a solução comparando o seu valor com os limites inferiores e superiores já encontrados e verificando se representa uma solução admissível do problema inicial. No fim de cada iteração, caso existam sub-problemas por resolver, é escolhido o próximo sub-problema a resolver, com base na estratégia estabelecida

### 3.1.2 *Branch-and-Cut*

Um outro método exato é o *Branch-and-Cut*(BeC) (LIEBERMAN, 2010),que é um algoritmo do tipo BeB, no qual são gerados planos de corte. Este método resolve sucessivamente problemas de programação linear em que se eliminam restrições de integralidade e por vezes restrições que complicam a resolução da relaxação linear. Em cada iteração obtém-se então uma solução ótima de uma relaxação linear que, não satisfazendo todas as restrições de integralidade, origina a geração de um novo corte, ou seja, de uma restrição que, eliminando a solução do problema anterior, não elimine a solução ótima do problema inicial. Pretende-se pois, encontrar novas restrições que são satisfeitas por todos os pontos admissíveis do problema original, mas são violados pela solução do problema corrente. O método de planos de corte aprimora, iterativamente, o conjunto de soluções através de desigualdades lineares de tal modo que a resolução do problema seguinte produza uma solução diferente que não viola as mesmas restrições de integralidade.

### 3.1.3 *Branch-and-Price*

Outro método exato, o Branch-and-Price(BeP), consiste na geração de colunas e de cortes, na regra de *branching*, na seleção de nós e limites superiores e, na escolha das colunas a introduzir nos sub-problemas, sendo aplicado a cada nó da árvore (BeB). Ou seja, no início do BeP são excluídas colunas por relaxação de modo a reduzir os requisitos de cálculo e de memória e, posteriormente, as colunas vão sendo adicionadas ao problema relaxado à medida que se torna necessário. Esta técnica é híbrida, pois combina os métodos de BeB e de Geração de Colunas.

(DELL?AMICO; RIGHINI; SALANI, 2006) propõe algoritmo do tipo BeP considerando quer programação dinâmica quer relaxação de espaço de estados (*State Space Relaxation*) para instâncias com 40 clientes. A relaxação de espaço de estados cria espaços de menor dimensão a ser explorados pelo algoritmo de programação dinâmica;

### 3.1.4 *Branch-and-Cut-and-Price*

O desenvolvimento de algoritmos enumerativos associados a métodos de geração de cortes (algoritmos *branch-and-cut*) e de geração de colunas (algoritmos *branch-and-price*) é um campo razoavelmente bem explorado. Contudo, desde que a geração de cortes e a de colunas foram estabelecidas como duas das técnicas mais importantes na programação inteira, tem-se procurado maneiras de combiná-las de forma eficiente em um mesmo algoritmo.

O algoritmo *branch-and-cut-and-price* é uma especialização do branch-and-bound em que novas colunas e novas desigualdades válidas são geradas dinamicamente à medida que a árvore de busca é percorrida. Embora este algoritmo utilize várias das técnicas usadas nos algoritmos *branch-and-cut* e *branch-and-price*(que essencialmente tem o mesmo princípio básico), o resultado dessa combinação requer técnicas muito mais sofisticadas do que as utilizadas em cada um em separado. Uma das razões é a necessidade de se acrescentar novas desigualdades (cortes) sem alterar a estrutura do subproblema de geração de colunas.

### 3.1.5 Resumo dos resultados para métodos exatos

No contexto PRVJT, a maneira mais comum de comparar a heurística são os resultados obtidos para os problemas de referência de (SOLOMON, 1987) 56. Esses problemas têm uma centena de clientes, um depósito central, restrições de capacidade, janelas de tempo no momento da entrega e uma restrição de tempo de rota total. As classes C1 e C2 têm clientes localizados em clusters e nas classes R1 e R2 os clientes estão em posições aleatórias. As classes RC1 e RC2 contêm uma mistura de clientes aleatórios e agrupados. Cada classe contém entre 8 e 12 instâncias de problemas individuais e todos os

problemas de uma classe têm as mesmas localizações do cliente e as mesmas capacidades do veículo; Apenas as janelas de tempo diferem. Em termos de densidade de janela de tempo (a porcentagem de clientes com janelas de tempo), os problemas têm janelas de tempo 25 %, 50 %, 75 %, e 100 %. Os problemas C1, R1 e RC1 têm um horizonte de agendamento curto, e exigem de 9 a 19 veículos. Problemas de horizonte curto têm veículos que têm capacidades pequenas e tempos de rota curtos, e não podem atender muitos clientes ao mesmo tempo. As classes C2, R2 e RC2 são mais representativas da entrega de "*long-haul*"com horários de programação mais longos e menos (2-4) veículos. Tanto o tempo de viagem como a distância são dados pela distância euclidiana entre os pontos. Resumo dos resultados por (AZI; GENDREAU; POTVIN, 2012).

| Instances | | JPSP08 | | DLH08 | | BMR11 | |
|---|---|---|---|---|---|---|---|
| Series | No. | No. Solved | Avg. Time | No. Solved | Avg. Time | No. Solved | Avg. Time |
| C1 | 9 | 9 | 468 | 9 | 18 | 9 | 25 |
| RC1 | 8 | 8 | 11,005 | 8 | 2,150 | 8 | 276 |
| R1 | 12 | 12 | 27,412 | 12 | 2,327 | 12 | 251 |
| C2 | 8 | 7 | 2,795 | 8 | 2,093 | 8 | 40 |
| RC2 | 8 | 5 | 3,204 | 6 | 15,394 | 8 | 3,767 |
| R2 | 11 | 4 | 35,292 | 8 | 63,068 | 10 | 28,680 |
| **Total** | 56 | 45 | | 51 | | 55 | |

Figura 5 – Resultados para métodos exatos

Para cada série de casos de 100 clientes, na Tabela de resultados obtidos por três algoritmos exatos , (JEPSEN; SPOORENDONK; RøPKE, 2013),(BENAVENT et al., 2014), e (BALDACCI; MINGOZZI; ROBERTI, 2011), abreviado por JSPS08, DLH08, e BMR11. Nesta tabela, as duas primeiras colunas indicam a série de instância e o número de instâncias que ele contém. Para cada série e cada algoritmo, informa o número de casos que foram resolvidos para otimização (sem limite de tempo imposto) e o tempo médio em segundos para resolvê-los. Estes tempos são os relatados pelos autores e foram obtidos em computadores com características diferentes: P4 de 3,0 GHz para JSPS08, AMD Opteron de 2,6 GHz para DLH08 e IBM Intel Xeon X7350 a 2,93 GHz para BMR11. A última linha da Tabela proporciona o número total de ocorrências resolvidos por cada algoritmo.

Estes resultados mostram claramente que os exemplos na classe 2 são muito mais difíceis de resolver do que os exemplos da classe 1, porque janelas de tempo de longas aumentam o número de caminhos possíveis e o número de clientes por caminho viável, obtendo instâncias mais difíceis de resolver . Os resultados também mostram a evolução

rápida dos algoritmos recentes. Antes do artigo de (JEPSEN; SPOORENDONK; RøPKE, 2013), apenas 35 dos 56 casos foram resolvidos. Com a introdução das desigualdades SR no algoritmo de *Branch-and-Cut-and-Price*, (JEPSEN; SPOORENDONK; RøPKE, 2013). elevou este número para 43. Com busca tabu e gerador de colunas, o *Branch-and-Cut-and-Price* de (BENAVENT et al., 2014) resolveu 51 instancias . Com uma abordagem baseada no conjunto modelo de particionamento reduzida e contando com n *g-paths* , (BALDACCI; MINGOZZI; ROBERTI, 2011) resolveram todos os casos, exceto um, devido a uma falta de memória. Os tempos de computação são muito menores do que os anteriores, mas deve notar-se que o seu método requer um limite superior.

No futuro, podemos esperar para ver mais resultados computacionais para as instâncias de referência (GEHRING. . . , 2016), que estendem as definidos para casos que envolvem 200, 400, 600, 800, e 1000 clientes . Para o melhor de nosso conhecimento (LARSEN, 2000),(BRäYSY; GENDREAU, 2002) (COOK WILLIAM & RICH, 1999), (KALLEHAUGE; LARSEN; MADSEN, 2006) Relatam resolver alguns destes casos.

## 3.2 HEURÍSTICAS

Heurísticas são métodos de solução que muitas vezes podem encontrar soluções viáveis de boa qualidade de forma relativamente rápida. Segundo (HILLIER; LIEBERMAN, 2005), o procedimento normalmente é um algoritmo iterativo completo em que cada iteração envolve a condução de uma busca por uma nova solução que, eventualmente, poderia superar o melhor resultado encontrado previamente. Quando o algoritmo termina após um tempo razoável, a solução por ele fornecida é a melhor que foi encontrada durante qualquer iteração. No entanto, não há nenhuma garantia em relação à qualidade solução. Heurísticas são, assim, testados empiricamente e seu desempenho é julgado por seus resultados computacionais. Atualmente, a maioria das VRPs encontrados na indústria são resolvidas usando heurísticas por causa de sua velocidade e sua capacidade de lidar com grandes instâncias. A tradição dita que uma função objetivo hierárquica é usado quando heurísticas são aplicadas: a prioridade é minimizar o número de veículos utilizados e o segundo objetivo é minimizar o custo dos caminhos percorridos. Os algoritmos exatos que não consideram o número de veículos na função objetivo (BRäYSY; GENDREAU, 2005).

As meta-heurísticas são uma classe de heurísticas mais recentes, que possuem como diferencial uma série de ferramentas que reduzem o risco de paradas prematuras em ótimos locais ainda distantes de um ótimo global. Geralmente estas ferramentas são componentes aleatórios inseridos durante a execução do algoritmo, que permitem que outras zonas de soluções sejam exploradas.

### 3.2.1 Avaliação das Heurísticas

A avaliação de qualquer método heurístico está sujeito à comparação de uma série de critérios que se relacionam com vários aspectos do desempenho do algoritmo. Exemplos de tais critérios são: tempo de execução, qualidade da solução, facilidade de implementação, robustez e flexibilidade (CORDEAU; MAISCHBERGER, 2012). Uma vez que os métodos heurísticos são, concebidos para resolver problemas do mundo real, a flexibilidade é uma consideração importante. Um algoritmo deve ser capaz de lidar facilmente com as mudanças no modelo, as restrições e a função objetivo. Quanto à robustez, não deve ser excessivamente sensível às diferenças nas características do problema: uma heurística robusta não deve funcionar mal em qualquer instância. Além disso, um algoritmo deve ser capaz de produzir boas soluções cada vez que é aplicado a uma determinada instância. Isso deve ser destacado, pois qualquer heurística não é determinística e contém alguns componentes aleatórios, como valores de parâmetros escolhidos aleatoriamente. A saída de execuções separadas desses métodos não-determinísticos sobre o mesmo problema na prática nunca é a mesma. Isso torna difícil analisar e comparar resultados. Usando apenas os melhores resultados de uma heurística não-determinística, como muitas vezes é feito na literatura, pode criar uma imagem falsa de seu desempenho real. Assim, consideramos que os resultados médios baseados em execuções múltiplas em cada problema constituem uma base importante para a comparação de métodos não determinísticos. Além disso, também seria importante relatar o pior desempenho dos casos. Discussões extensas sobre esses assuntos podem ser encontradas em (CORDEAU; MAISCHBERGER, 2012).

O tempo que uma heurística leva para produzir soluções de boa qualidade pode ser crucial ao escolher entre diferentes técnicas. Da mesma forma, a qualidade da solução final, medida pela função objetivo, é importante. Como a solução está próxima da solução ótima é uma medida padrão de qualidade ou, se a heurística é projetada para simplesmente produzir soluções viáveis, então a capacidade da heurística para fornecer essas soluções é importante.

Geralmente há um *trade-off* entre o tempo de execução e a qualidade da solução - quanto maior o tempo que uma heurística é executada, melhor a qualidade da solução final. Um compromisso é necessário para que as soluções de boa qualidade que são produzidas em um período razoável de tempo. Basicamente, esse *trade-off* entre tempo de execução e qualidade da solução pode ser visto em termos de uma otimização multi objetiva em que os dois objetivos são equilibrados. As medidas de desempenho para heurísticas podem ser plotadas em espaço bidimensional, com a primeira dimensão correspondente ao tempo de execução e A segunda para a qualidade da solução. Nesse espaço, pontos que não existem outros pontos com valores melhores em ambas as dimensões são considerados ótimos de Pareto; Eles definem compromissos efetivos entre os objetivos. (RUSSELL; NORVIG, 1994) e (BRäYSY; GENDREAU, 2005) . A escolha entre diferentes abordagens heurísticas

produzindo resultados ótimos de Pareto depende das preferências do tomador de decisão e da situação em questão.

O método mais comum para avaliar a qualidade da solução de um algoritmo heurístico é a análise empírica. Em geral, a análise empírica envolve o teste da heurística em uma ampla gama de instâncias de problema para ter uma ideia do desempenho geral. Para se chegar a conclusões que tenham significado num sentido estatístico, o desenho experimental deve idealmente ser usado em diferentes níveis dos vários parâmetros do algoritmo e os resultados comparados por técnicas apropriadas.

Dificuldades enfrentadas especialmente no contexto PRVJT são que muitas vezes apenas os melhores resultados obtidos durante todo o estudo computacional são relatados. Além disso, em alguns casos os autores não relatam o número de execuções ou tempo de CPU necessário para obter os resultados relatados. Nestes casos é impossível concluir qualquer coisa sobre a eficiência dos métodos, ou comparar estes métodos com outras abordagens. A única base adequada para a comparação destes métodos seriam soluções ótimas, uma vez que se houver tempo suficiente disponível, é sempre preferível resolver os problemas com a optimização utilizando métodos exatos. Para ser capaz de chegar a conclusões apropriadas, além do número de execuções e consumo de tempo, deve-se responder a perguntas como quais são os limites do algoritmo dado, ou seja, quão bons são os melhores resultados que podem ser obtidos usando a abordagem particular , E como uma boa solução pode ser obtida em uma determinada quantidade de tempo. Em outras palavras, deve-se relatar resultados obtidos usando diferentes tempos de computação e observar quanto tempo é necessário para obter resultados de uma determinada qualidade. Além disso, na nossa opinião, os números que descrevem a relação entre a qualidade da solução e o tempo de computação facilitariam muito a análise. (TAILLARD, 2001) discute extensivamente esta questão e propõe relatar um esforço computacional absoluto, como o número de iterações em vez de computar tempos e usar diagramas de probabilidade baseados em testes estatísticos repetidos de Mann-Whitney. Obviamente, tal abordagem não é possível quando se confia em resultados previamente publicados como fazemos aqui.

Os resultados são geralmente classificados de acordo com uma função objetivo hierárquica, onde o número de veículos é considerado o objetivo primário, e para o mesmo número de veículos, o objetivo secundário é muitas vezes a distância total percorrida ou a duração total das rotas. Portanto, uma solução que requer menos rotas é sempre considerada melhor do que uma solução com mais rotas, independentemente da distância total percorrida. De acordo com (BRäYSY, 2001), esses dois objetivos são muitas vezes conflitantes, o que significa que a redução no número de veículos frequentemente causa aumento na distância total percorrida. Assim, uma melhor solução em termos de distância total pode ser obtida aumentando o número de rotas. Alguns outros artigos relatam achados semelhantes, ver, por exemplo,(YVES; LABURTHE, 1999)

### 3.2.2 Heurísticas Para Construção de Rotas

Heurísticas para construção de rotas seleciona nós (ou arcos) sequencialmente até que uma solução viável tenha sido criada. Os nós são escolhidos com base em algum critério de minimização de custos, muitas vezes sujeitos à restrição de que a seleção não cria uma violação da capacidade do veículo ou restrições de janela de tempo. Métodos sequenciais construir uma rota de cada vez, enquanto que os métodos paralelos construir várias rotas simultaneamente.

(SOLOMON, 1987) propõe um esquema denominado rota-primeiro cluster-segundo usando uma heurística *Giant-Tour*. Primeiro, os clientes são agendados em um *Giant-Tour* e, em seguida, este *tour* é dividido em um número de rotas menores. Um *Giant-Tour* inicial poderia, por exemplo, ser gerado como um caixeiro viajante sem considerar as restrições de capacidade e tempo.

(SOLOMON, 1987) descreve várias heurísticas para o PRVJT. Um dos métodos é uma extensão da heurística de poupança de Clarke e Wright (1964). O método de poupança, originalmente desenvolvido para a VRP clássica, é provavelmente a heurística de construção de rotas mais conhecida. Começa com uma solução em que cada cliente é fornecido individualmente por uma rota separada. Combinando as duas rotas que servem respectivamente os clientes i e j resulta em uma economia de custos de $S_\gamma = d_{i0} + d_{0j} - d_\gamma$ Clarke e Wright (1964) selecionam o arco (i, j) ligando Clientes i e j com máximo $S_{ij}$ sujeitos à exigência de que a rota combinada seja viável. Com esta convenção, a operação de combinação de rotas é aplicada iterativamente. Ao combinar rotas, pode simultaneamente formar rotas parciais para todos os veículos ou adicionar sequencialmente clientes a uma determinada rota até que o veículo esteja totalmente carregado. Para ter em conta a proximidade espacial e temporal dos clientes, a (SOLOMON, 1987) estabelece um limite para o tempo de espera da rota.

A segunda heurística, uma vizinha mais próxima do tempo, inicia cada rota encontrando um cliente não roteado mais próximo do depósito. Em cada iteração subsequente, a busca heurística para o cliente mais próximo do último cliente adicionado na rota e adiciona-lo no final da rota. Uma nova rota é iniciada sempre que a pesquisa não consegue encontrar um local de inserção viável, a menos que não haja mais clientes não roteados. A métrica usada para medir a proximidade de qualquer par de clientes tenta explicar a proximidade geográfica e temporal dos clientes. A métrica usada para medir a proximidade de qualquer par de clientes tenta explicar a proximidade geográfica e temporal dos clientes.

A mais bem sucedida das três heurísticas de inserção sequencial propostas é chamada I1. Inicialmente, uma rota é inicializada com um cliente "semente"e os clientes não roteados restantes são adicionados a esta rota até que ela esteja cheia em relação ao horizonte de agendamento e / ou restrição de capacidade. Se os clientes não roteados permanecerem, as inicializações e os procedimentos de inserção serão repetidos até que todos os clientes sejam

atendidos. Os clientes de semente são selecionados encontrando o cliente geograficamente mais distante não roteado em relação ao depósito ou o cliente não roteado com o menor tempo de início permitido para o serviço.

## 3.3 BUSCA LOCAL

Um conceito central na maioria das heurísticas de sucesso para o PRVJT é a de busca local. Algoritmos de busca local são baseados em vizinhanças . Seja $\theta$ um conjunto de soluções viáveis para uma determinada instância PRVJT, e deixe c: $\theta \rightarrow$ R ser uma função que mapeia a partir de uma solução para o custo para esta solução. O conjunto $\theta$ é finito, mas muitas vezes é extremamente grande. Desde o PRVJT é um problema de minimização, o nosso objectivo é encontrar uma solução s* para o qual c(s*) $\leq$c(s) para todo s $\in \theta$ . No entanto, com a heurística, estamos dispostos a se contentar com uma solução que pode ser ligeiramente inferior ao s*.

Seja P($\theta$) o conjunto de subconjuntos de soluções em ($\theta$). Definimos uma função de vizinhança como uma função N:$\theta \rightarrow$ P($\theta$) P(S) que mapeia a partir de uma solução de s para um subconjunto de soluções de N(s). Este subconjunto é chamado o vizinho de s. Uma solução s é dito ser localmente ótima ou uma condição local no que diz respeito a uma vizinhança N(s) se c(s) $\leq$ C($s^i$) de todos $s^i \in$ N(s). Com estas definições, podemos descrever um algoritmo decida acentuada(*steepest descent*) (veja Algoritmo 3.1). O algoritmo leva uma solução s inicial como entrada. Em cada iteração, ele encontra a melhor solução $s^i$ na vizinhança N(s) da solução corrente s (linha 4). Se $s^i$ é melhor do que s (linha 5)

**Algorithm 5.1** Steepest descent

```
1: input: Initial solution s ∈ 𝒮
2: done = false
3: while done ≠ true do
4:     s′ ∈ arg min_{s″∈N(s)} {c(s″)}
5:     if c(s′) < c(s) then
6:         s = s′
7:     else
8:         done = true
9: return s
```

Figura 6 – algoritmo *steepest descent*

em seguida,$s^i$ substitui s como a solução atual (linha 6). Linhas 3-8 são repetidos enquanto $s^i$ melhorar a solução. Quando o loop parar, o algoritmo retorna s como a melhor solução encontrada. O algoritmo é chamado um algoritmo de descida acentuada, porque sempre escolhe a melhor solução na vizinhança atual.

Em geral,grandes vizinhanças conduzem a melhores soluções, quando as vizinhanças são utilizados, por exemplo, num algoritmo de descida mais acentuada. A desvantagem de grandes vizinhanças é que a avaliação de todas as soluções é mais demorada. Analisamos uma série de vizinhos que foram utilizados nas heurísticas PRVJT mais bem sucedidos. Classificamos esses vizinhos em duas categorias: vizinhos tradicionais e grandes. A primeira categoria engloba os vizinhos, cujo tamanho está crescendo polinomialmente com n de uma forma controlada de tal modo que todas as soluções na vizinhança podem ser avaliada por enumeração explícita.Vizinhos de tamanho cujo crescimento é tão rápida que não pode ser pesquisado explicitamente , e vizinhos cujo tamanho está a crescer exponencialmente (AHUJA; ORLIN; SHARMA, 2000).

### 3.3.1 2-opt

Contém soluções obtidas através da remoção de dois arcos de uma rota e substituí-los por outros dois arcos para reconectar o percurso, ao alterar a orientação do subcaminhos que não contém o depósito. Na figura a praça representa o depósito e os círculos são clientes. Os arcos tracejadas correspondem a subcaminhos em G (ver Secção 3.2), envolvendo um ou vários arcos, enquanto que cada arco sólida corresponde a um único arco em G. Observe que alterar a orientação do subcaminhos (i + 1,.,.,. j + 1) pode ser problemática devido às janelas de tempo.

### 3.3.2 Or-opt

Cada solução no vizinho or-opt é definido pela relocação de um subcaminho (i + 1,..., J) para uma posição diferente na rota. A orientação do subcaminhos realocado é preservada. O movimento é realizado através da remoção de arcos (i, i + 1), (j, j + 1), e (k, k + 1) e substituindo-os pelos arcos (i, j + 1), (k, i + 1 ), e (j, k + 1). Normalmente, o vizinho é reduzido, considerando apenas subcaminhos contendo um número limitado de clientes ou tentando apenas inserções que estão perto de sua posição original. (BRäYSY; GENDREAU, 2005).

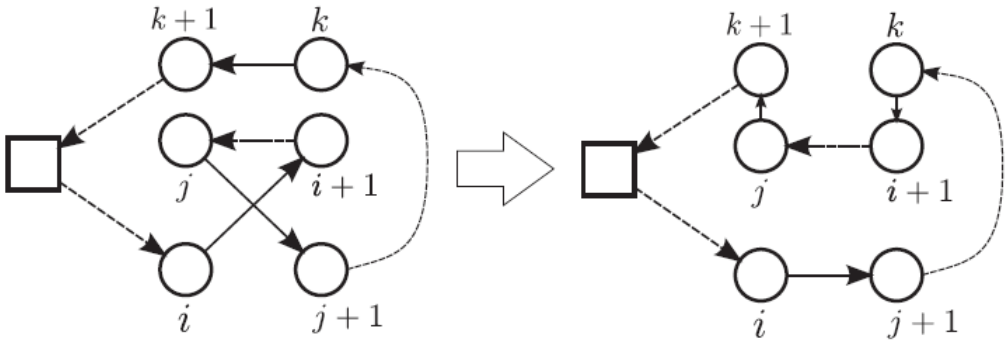


Figura 7 – Or-opt

### 3.3.3 2-opt*

O 2-opt* é definido de forma semelhante ao 2-opt , mas as suas soluções são derivadas modificando duas rotas em vez de uma. Dois arcos, (i, i + 1) e (j, j + 1), a partir de duas rotas distintas são removidos, e as rotas são restabelecida através da inserção dos arcos (i, j+1) e (j, i+1) . O efeito é que as extremidades das duas rotas são trocados. O 2-opt* não altera a orientação dos subcaminhos, em oposição ao 2-opt.

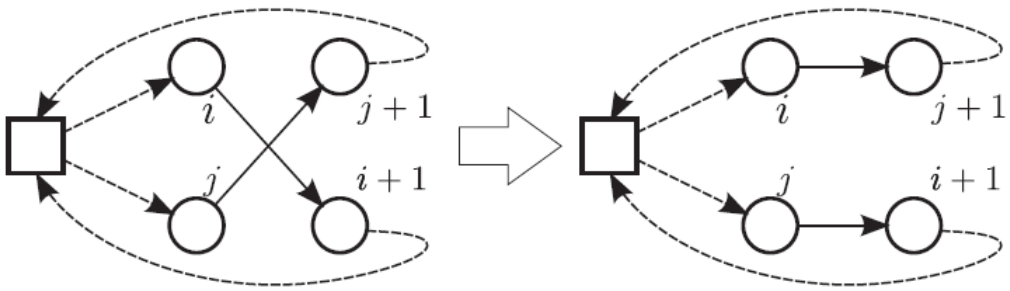


Figura 8 – 2-opt*

### 3.3.4 *Cross Exchange*

No vizinho de intercâmbio, dois subcaminhos são selecionados e as suas posições são trocadas. Isto é feito através da remoção de quatro arcos (i, i + 1), (j, j + 1), (k, k + 1), e (L, L + 1) e substituindo-os pelos arcos (I, K + 1), (L, J + 1), (K, i + 1), e (j, l + 1). O tamanho da vizinhança de troca cruzada é tipicamente reduzida por considerar apenas subcaminhos que contêm um número limitado de clientes. Notamos que *Cross Exchange* pode ser utilizado de forma intra-percurso onde os subcaminhos para ser trocado pertencem à mesma via.

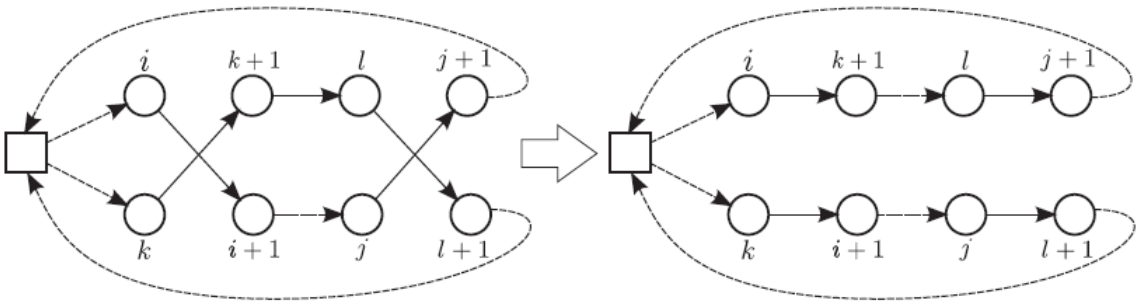


Figura 9 – Cross Exchange

### 3.3.5 Realocação de Caminho

No caminho de realocação , as soluções são obtidas através da relocação de um subcaminho de uma rota para outra. Isto é feito através da remoção de três arcos (i, i

+ 1), (j, j + 1), e (k, k + 1) e substituindo-os pelos arcos (i, j +1), (J, K + 1), e (k, i + 1). Esta zona pode ser visto como um caso especial de vizinhança de troca cruzada, se permitirmos que um subcaminhos vazio no último vizinhança. O comprimento do subcaminhos realocada é tipicamente limitado, e algumas implementações permitir que o subcaminhos de ser invertida quando é reinserido.

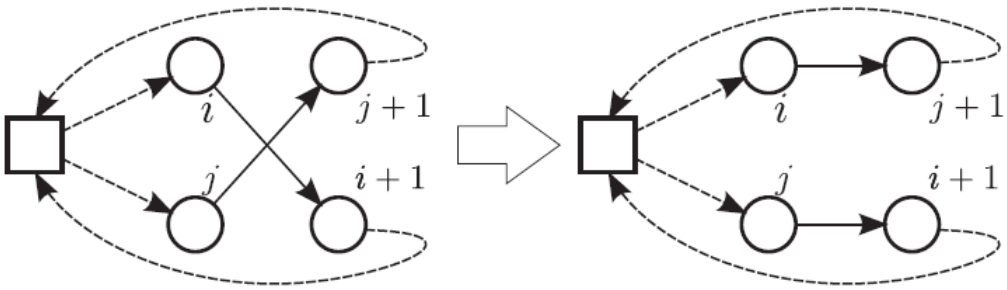


Figura 10 – Realocação de Caminho

### 3.3.6 Técnicas de aumento de velocidade

Em busca local, é importante para implementar a avaliação vizinho de forma eficiente, a fim de procurar tantas soluções quanto possível dentro do tempo previsto para a pesquisa. Cada movimento nos vizinhos mencionados acima podem ser avaliadas em tempo constante utilizando as técnicas descritas em (KINDERVATER; SAVELSBERGH, 1997), mas para grandes instâncias isso pode não ser suficiente. Por isso, é comum a truncar a busca na vizinhança de uma maneira heurística de tal forma que alguns movimentos que não parecem promissores não são avaliados. Isto pode ser feito considerando-se apenas movimentos que ligam os clientes próximos; considerações de janela de tempo pode também ser tida em conta. Dois exemplos de tal filtragem pode ser encontrado em (UNO; YAGIURA, 2000) , (NAGATA; BRÄYSY; DULLAERT, 2010).

(Daniele Vigo; Paolo Toth, 2014) introduziu busca sequencial para o PRVJT. busca sequencial é capaz de acelerar a busca local usando a filtragem exato onde grandes subconjuntos dos movimentos possíveis pode ser ignorada com base em considerações de custo. resultados computacionais mostram speedups impressionantes para instâncias PRVJT em grande escala. pesquisa sequencial tem, com o melhor de nosso conhecimento, não foi ainda utilizada nos PRVJT mais bem sucedidos.

### 3.3.7 Permitindo Soluções inviáveis

Ao projetar meta-heurísticas para a PRVJT, deve-se decidir se soluções inviáveis deve ser permitido durante a pesquisa. Quando permitido, eles são tipicamente penalizada na função objetivo por um ou mais termos que visam reduzir inviabilidade. Permitindo soluções inviáveis torna mais fácil de manobrar no espaço de solução, fornecendo atalhos

entre as áreas de soluções de alta qualidade. Por outro lado, aumenta a complexidade do algoritmo: avaliação do objetivo pode ser mais difícil, parâmetros de penalidade deve ser ajustado (potencialmente de um modo adaptativo), e deve-se assegurar que as soluções viáveis são visitadas pelo menos de vez em quando. Apesar destes inconvenientes, permitindo soluções inviáveis parece ser uma ferramenta importante para encontrar soluções de alta qualidade.

Nas meta-heurísticas atuais são permitidos três tipos de soluções inviáveis : janela de tempo, capacidade do veículo, e as violações de atendimento ao cliente. O último tipo ocorre quando alguns clientes não são visitados. É o tipo mais comum nos algoritmos que minimizam o número de veículos utilizados. violações de janelas e de capacidade de tempo são normalmente autorizados juntos e levar a um c(s) função objetivo penalizada.

Em que c(s) é a função de objetivo padrão e q(s) (resp., w(s)) mede a capacidade total (resp., tempo-janela) violação sobre todas as rotas. E são parâmetros que são geralmente ajustados durante a pesquisa, reagindo ao desempenho do algoritmo. Um bom exemplo de como a parâmetros e pode gestão é dado em (CORDEAU; LAPORTE; MERCIER, 2001).

Violação de uma janela de tempo ao longo de uma rota r = (... $v_0 = 0$, $v_1$, $v_2$,..., $v_k$, $v_{k+1}$ = n + 1) é calculada diretamente: o início do tempo de serviço $T_{V_I}$ no vértice $v_i$ é calculado como

e a violação total é de $\sum_{k+1}^{i=1}$max0, $T_{v_i}$ - $b_{v_i}$. A desvantagem deste procedimento é que a avaliação de um movimento nos vizinhos já não pode ser realizada em tempo constante. Em vez disso, (NAGATA; BRÄYSY; DULLAERT, 2010) propôs um início artificial do avaliação do tempo de serviço que se move do início para o fim da janela de tempo sempre que for violada.

Violação da janela de tempo é dada por $\sum_{k+1}^{i=1}$max0, $T_{v_i}^i$ - $b_{v_i}$, uma fórmula que ainda captura violações de janela de tempo e faz com que seja possível avaliar em tempo constante um único movimento nos vizinhos (GENDREAU; POTVIN, 2010).

### 3.3.8 Minimizar o Numero de Veículos Usados

É tipicamente executada em duas fases, onde uma primeira procura encontrar o menor número de veículos necessários e, em seguida, mantém este número fixo enquanto minimiza o custo total da viagem. Se a heurística permite visitar soluções inviáveis, então pode-se selecionar um número de veículos m e criar uma solução (possivelmente inviável) com esse número de veículos. Em seguida, uma pesquisa é realizada com o objetivo de encontrar uma solução viável. Se a pesquisa for bem sucedida, o número de veículos é reduzida no excluindo uma rota completa antes de repetir a busca. Caso contrário, o número de veículos é aumentado no e a pesquisa é repetida ,se ainda não houver soluções

viáveis encontradas . Usando um limite inferior calculado sobre o número de veículos, é possível parar a busca de um menor número de veículos, quando o limite inferior é atendido. A pesquisa para o número mínimo de veículos também pode ser abortado quando um esforço suficiente foi investido. (NAGATA; BRÄYSY; DULLAERT, 2010) criaram um do procedimento rota minimização mais eficaz.

### 3.3.9 Pesquisa Caminho Único

Parte de uma única solução, algoritmos de busca caminho único gerar uma sequência de soluções que podem ser vistos como uma trajetória através do espaço de solução. Em cada iteração, apenas a solução de corrente é utilizada para determinar a próxima.

(HASHIMOTO; YAGIURA; IBARAKI, 2008a) desenvolveram um algoritmo de busca local iterado por uma variante do PRVJT onde as janelas de tempo são representados por funções de penalidade linear convexa, por partes. O PRVJT é um caso especial deste problema quando a função de penalidade é definida de forma adequada (penalidade infinita fora da região viável). Por causa das funções de penalidade, é não-trivial para determinar o melhor momento de partida possível para uma determinada rota. Esse problema é chamado o *Optimal Start Time Problem* (OSTP). Os autores apresentam um algoritmo de programação dinâmica para a OSTP e descrever como o algoritmo de programação dinâmica pode ser acelerado no algoritmo de busca local. Note que o OSTP foi previamente estudado por (DESROCHERS et al., 1988), que considerou funções de penalidade convexas arbitrárias.

(HASHIMOTO; YAGIURA; IBARAKI, 2008a) propôs um algoritmo de busca local iterado para uma PRVJT dependente do tempo com janelas de tempo suaves por busca local iterativa. Como antes, é uma tarefa não trivial para determinar o tempo de partida de uma rota ótima. Quando um ótimo local é encontrado, a solução atual é perturbado pela aplicação de um único movimento de troca aleatório para a solução. O algoritmo é usado para resolver instâncias PRVJT, permitindo violações janela de tempo e ajustar penalidades de forma adequada.

(CORDEAU; MAISCHBERGER, 2011) introduziu um algoritmo de busca local iterado que se baseia na versão atualizada do algoritmo de busca tabu de (CORDEAU; LAPORTE; MERCIER, 2004), como o algoritmo de busca local. O algoritmo de busca tabu permite soluções que são inviáveis no que diz respeito às duas janelas de tempo e capacidade do veículo. Ele usa uma vizinhança simples, baseada na realocação do cliente. O passo perturbação é inspirado pelo LNS e consiste em remover um conjunto de clientes e reinserindo-os na ordem aleatória. Cada cliente é reinserido na rota utilizando uma política *cheapest-insertion*.

| Author | R1 | R2 | C1 | C2 | RC1 | RC2 | CNV/CTD |
|---|---|---|---|---|---|---|---|
| (1) Thompson et al. (1993) | **13.00** | **3.18** | **10.00** | **3.00** | **13.00** | **3.71** | **438** |
| | 1356.92 | 1276.00 | 916.67 | 644.63 | 1514.29 | 1634.43 | 68916 |
| (2) Potvin et al. (1995) | **13.33** | **3.27** | **10.00** | **3.13** | **13.25** | **3.88** | **448** |
| | 1381.9 | 1293.4 | 902.9 | 653.2 | 1545.3 | 1595.1 | 69285 |
| (3) Russell (1995) | **12.66** | **2.91** | **10.00** | **3.00** | **12.38** | **3.38** | **424** |
| | 1317 | 1167 | 930 | 681 | 1523 | 1398 | 65827 |
| (4) Antes et al. (1995) | **12.83** | **3.09** | **10.00** | **3.00** | **12.50** | **3.38** | **429** |
| | 1386.46 | 1366.48 | 955.39 | 717.31 | 1545.92 | 1598.06 | 71158 |
| (5) Prosser et al. (1996) | **13.50** | **4.09** | **10.00** | **3.13** | **13.50** | **5.13** | **471** |
| | 1242.40 | 977.12 | 843.84 | 607.58 | 1408.76 | 1111.37 | 58273 |
| (6) Shaw (1997) | **12.31** | ____ | **10.00** | ____ | **12.00** | ____ | ____ |
| | 1205.06 | | 828.38 | | 1360.40 | | |
| (7) Shaw (1998) | **12.33** | ____ | **10.00** | ____ | **11.95** | ____ | ____ |
| | 1201.79 | | 828.38 | | 1364.17 | | |
| (8) Caseau et al. (1999) | **12.42** | **3.09** | **10.00** | **3.00** | **12.00** | **3.38** | **420** |
| | 1233.34 | 990.99 | 828.38 | 596.63 | 1403.74 | 1220.99 | 58927 |
| (9) Schrimpf et al. (2000) | **12.08** | **2.82** | **10.00** | **3.00** | **11.88** | **3.38** | **412** |
| | 1211.53 | 949.27 | 828.38 | 589.86 | 1361.76 | 1097.63 | 56830 |
| (10) Cordone et al. (2001) | **12.50** | **2.91** | **10.00** | **3.00** | **12.38** | **3.38** | **422** |
| | 1241.89 | 995.39 | 834.05 | 591.78 | 1408.87 | 1139.70 | 58481 |
| (11) Bräysy (2001a) | **12.17** | **2.82** | **10.00** | **3.00** | **11.88** | **3.25** | **412** |
| | 1253.24 | 1039.56 | 832.88 | 593.49 | 1408.44 | 1244.96 | 59945 |

(1) PC/AT 12 MHz, 4 runs, 1.8 min., (2) Sparc workstation, –, 3.0 min., (3) PC/486/DX2 66 MHz, 3 runs, 1.4 min., (4) RS6000/530, 4 runs, 3.6 min., (5) –,–,– , (6) DEC Alpha, 3 runs, 2 hours, (7) Sun Ultra Sparc 143 MHz, 6 runs, 1 hour, (8) Pentium 300 MHz, 1 run, 5 min., (9) RS 6000, –, 30 min., (10) Pentium 166 MHz, 1 run, 15.7 min., (11) Pentium 200 MHz, 1 run, 4.6 min.,

Figura 11 – Resultados obtidos com as heurísticas de busca local

## 3.4 LARGE NEIGHBORHOOD SEARCH

O LNS heurística original por (SHAW, 1998) já resolveu o PRVJT com bons resultados no conjunto reduzido de casos, mas foi só com o trabalho de (BENT; HENTENRYCK, 2004) que a capacidade do método para a obtenção de boas soluções para o PRVJT foi totalmente revelado.A pesquisa funciona escolhendo de forma aleatória um conjunto de visitas de clientes. Os clientes selecionados são removidos da programação e, em seguida, reinseridos com o custo ótimo. Para criar a oportunidade para o intercâmbio de visitas de cliente entre rotas, as visitas removidas são escolhidas de modo que estejam relacionadas. Aqui, o termo relacionado refere-se a clientes que estão geograficamente próximos um do outro, servidos pelo mesmo veículo, têm uma procura semelhante de mercadorias e tempos de início semelhantes para o serviço. Um ramo e método ligado acoplado com a Programação de Restrição é então usado para reprogramar visitas removidas. Na solução inicial, cada cliente é servido por um veículo separado. Devido aos elevados requisitos computacionais, esta abordagem só pode ser aplicada a problemas em que o número de clientes por rota é relativamente baixo.

## 3.5 BUSCA TABU

A meta-heurística Busca Tabu (BT) é uma técnica que segue os princípios gerais de Inteligência Artificial (IA) e é utilizada para guiar o procedimento de busca local na busca

do ótimo do problema. BT toma da IA o conceito de memória e a implementa mediante estruturas simples com o objetivo de conduzir a busca considerando o histórico da mesma. O procedimento trata de extrair informação do passado recente e atuar em função dessa informação. Neste sentido pode-se dizer que existe um certo aprendizado e que a busca é inteligente.

Busca Tabu tem suas raízes na década de 60, mas a forma como hoje é apresentada foi proposta por (GLOVER; LAGUNA, 1997). BT é uma abordagem plenamente aceita para o tratamento de problemas de otimização combinatória e tem se mostrado como uma técnica muito promissora através de inúmeras aplicações bem sucedidas.

O método é uma técnica iterativa que explora o conjunto de soluções de um problema, movimentando-se de uma solução S para outra solução S*, numa determinada vizinhança N(S) de S. Esses movimentos são executados com o intuito de alcançar eficientemente uma solução qualificada como boa, que pode ser ótima ou próxima da ótima. Ao contrário de um algoritmo simples de descida, BT aceita soluções S*piores que S, evitando assim o problema de ótimo local, fazendo uso de uma estrutura de memória capaz de suportar e até encorajar uma busca não monotonicamente decrescente (em problemas de minimização). BT percorre o espaço de busca, analisando os movimentos através de regras determinísticas. Quando BT aceita movimentos que pioram o valor da função objetivo, podem ocorrer ciclos, isto é, o retorno a soluções já visitadas. Para evitar que ciclagens se estabeleçam, BT armazena os atributos das soluções já pesquisadas em uma lista chamada lista tabu. Os atributos armazenados são muitas vezes usados para impor condições, chamadas restrições tabu, que impedem a escolha de movimentos indesejáveis. Um movimento é chamado tabu se conduz a soluções cujos atributos estão contidos na lista tabu ou satisfazem restrições tabu. Após um determinado número de iterações a lista tabu é alterada.

Uma estrutura de geração de vizinhança de boa qualidade é fundamental para o sucesso da técnica. Se um movimento provoca uma variação muito pequena na função objetivo, é razoável supor que a busca por boas trajetórias para escapar de um mínimo local será problemática. Por outro lado, se essa amplitude é grande, certamente haverá dificuldades para encontrar um mínimo local cuja qualidade seja próxima do mínimo global.

Existem alguns refinamentos que podem ser implementados no procedimento padrão de BT de forma a torná-lo mais eficiente. Esses refinamentos tentam introduzir mais inteligência no processo de busca. Um deles se refere ao tamanho da lista tabu, que varia de acordo com o problema em estudo. O tamanho da lista depende também das restrições tabu empregadas. Em geral, restrições fortes implicam em listas pequenas. Uma maneira de identificar-se o tamanho da lista apropriada é simplesmente observar a ocorrência de ciclos (listas muito pequenas) e a deterioração da qualidade da solução (listas muito grandes). A melhor lista certamente estará entre esses extremos. Outro aprimoramento que

se relaciona com o tamanho da vizinhança é a construção de uma lista de candidatos. Um exame completo de vizinhança geralmente proporciona soluções de boa qualidade, porém pode consumir muito tempo computacional. Por essa razão é importante a utilização de estratégias que isolam regiões da vizinhança contendo movimentos desejáveis e os colocam em uma lista de candidatos. Existem estudos sobre técnicas de decomposição de vizinhança, construção de listas de movimentos promissores, listas dos atributos preferidos e muitos outros.

Uma implementação bastante atraente consiste na variação da penalidade imposta à função objetivo para destruir a estrutura de otimalidade local da qual se deseja escapar. A variação da penalidade é exemplo de um procedimento conhecido como oscilação estratégica , que representa uma das abordagens básicas de diversificação da busca tabu. A oscilação pode cruzar os limites e penetrar em regiões de inviabilidade ou somente se aproximar e recuar, sem transpor esses limites. Uma maneira simples de se implementar essa penalidade é alternar uma série de movimentos que melhoram a função objetivo com outra série de movimentos que a pioram.

## 3.6 PESQUISAS EM POPULAÇÕES

Meta-heurísticas que são baseadas na ideia de manter uma pool de soluções, chamada de população, que evolui a cada iteração do processo de solução. Ao contrário de pesquisa única trajetória, novas soluções são derivados a partir de uma população de soluções que oferece diversidade em si. Abaixo, vamos examinar as recentes obras sobre o PRVJT baseados em algoritmos evolutivos religação e trajeto, duas famílias de heurísticas de busca populacional.

### 3.6.1 Algoritmos evolutivos

Algoritmos evolutivos combinam soluções da população atual para produzir uma nova população. O mais adaptado deles é então retido para atualizar a população. A meta-heurística de Algoritmos Genéticos (AG) foi introduzida por (HOLLAND, 1975) inspirado no processo observado da evolução natural dos seres vivos, tentando imitar os mecanismos de reprodução existentes na natureza. Os AG estabelecem uma analogia entre o conjunto de soluções de um problema e o conjunto de indivíduos de uma população, codificando a informação de cada solução num vetor chamado de cromossomo. Para isso se introduz uma função de avaliação dos cromossomos, isto é a função objetivo (função aptidão). Igualmente se introduz um mecanismo de seleção de tal modo que os cromossomos com melhor avaliação sejam escolhidos para que se reproduzam.

Quando se trabalha com os AG tem-se que levar em consideração a necessidade de se integrar e implementar eficientemente duas ideias fundamentais: as representações simples

como vetores das soluções do problema e a realização de transformações simples para modificar e melhorar essas soluções. Para implementar os AG tem-se que considerar vários elementos principais tais como: a representação do cromossomo, uma população inicial, a função que avalia o cromossomo, os critérios de seleção e eliminação do cromossomo, uma ou varias operações de recombinação ( crossover ), uma ou varias operações de mutação .

Em geral a população inicial é gerada aleatoriamente. No entanto, ultimamente se estão utilizando métodos heurísticos para gerar soluções iniciais com boa qualidade. Neste caso é importante garantir a diversidade estrutural das soluções para evitar a convergência prematura. Em relação à avaliação dos cromossomos em geral utiliza-se o valor da função objetivo do problema.

A operação da mutação mais simples e uma das mais utilizadas consiste em substituir com certa probabilidade o valor de um bit. O papel que cumpre a mutação é de introduzir um fator de diversificação, pois em determinadas ocasiões a convergência do procedimento a boas soluções pode ser prematura e ficar preso em ótimo locais. Outra forma de introduzir novos elementos numa população é combinar elementos tomados aleatoriamente sem considerar a função aptidão.

**Algorithm 5.4** Memetic algorithm

1: Create an initial pool $P$ of solutions
2: **while** stopping criterion is not met **do**
3: $S = \text{crossover}(P)$
4: $S' = \text{mutation}(S)$
5: $S'' = \text{localsearch}(S')$
6: $P = \text{updatepool}(P, S'')$
7: return best solution observed during the search

Figura 12 – Algorítimos meméticos

Algoritmos meméticos hibridizam algoritmos genéticos com busca local e potencialmente algoritmos específicos de problemas. O tamanho da população inicial criada na linha 1 é, obviamente, um parâmetro importante. Linhas 2 a 6 constituem o ciclo principal do algoritmo que pode ser parado de acordo com vários critérios com base no número de iterações realizada, o tempo decorrido, ou uma medida da convergência. Na linha 3, novas soluções, formando conjunto S são construídos através da combinação de soluções existentes de P. Esta operação é chamado de crossover. Na linha 4, a mutação é aplicado a um subconjunto das soluções em C para criar um novo subconjunto de indivíduos $S^i$, perturbando deste modo a definir prole. O passo de mutação não está incluído em todos os algoritmos meméticos. Na linha 3. a busca local é executado em cada solução prole em $S^i$ para gerar um conjunto $S^{ii}$ de soluções melhoradas. Às vezes, esta etapa de busca local é

visto como uma operação de mutação. As soluções em $S^{ii}$ são então usadas para atualizar a população P (linha 6). Uma maneira simples de efetuar esta atualização consiste em manter as melhores soluções $\|P\|$ do conjunto P ($S^{ii}$, mas um processo mais avançado é normalmente usado para favorecer a diversidade da população P. Um algoritmo genético clássico é obtido através da remoção da etapa de busca local (linha 5) no Algoritmo

Um dos Componentes vitais em algoritmos meméticos é o algoritmo *crossover*. Abaixo damos uma visão geral dos métodos de crossover utilizados nos algoritmos evolutivos de maior sucesso para a PRVJT.

### 3.6.2 *Crossover* EAX

O algoritmo de *crossover* EAX é baseado no algoritmo de *crossover* forte para o TSP, que foi introduzido pela primeira vez por (NAGATA, 2006) e tem sido a base dos melhores algoritmos evolucionários para o TSP.

### 3.6.3 *Crossover* OX

OX crossover é um crossover clássico para representações de soluções à baseadas em permutações (FALKENAUER; BOUFFOUIX, 1991). Figura ilustra esse crossover para permutações dos números de 1 a 9. Um escolhe dois números aleatórios i e j e cria uma permutação para o individuo que vai ser gerado, copiando o pedaço de pai 1 a partir da posição i para j na posição i para j da prole. A parte restante da prole é encontrado copiando elementos de pai 2, respeitando a ordem de pai 2.

| position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Parent 1 | 4 | 3 | 8 | 2 | 9 | 6 | 5 | 7 | 1 |
| Parent 2 | 5 | 7 | 3 | 2 | 8 | 4 | 9 | 1 | 6 |
| offspring | 3 | 4 | 8 | 2 | 9 | 6 | 1 | 5 | 7 |

Figura 13 – Ox *crossover*

### 3.6.4 Religação de Caminhos

(HASHIMOTO; YAGIURA; IBARAKI, 2008b) desenvolveram um algoritmo de religação caminho para o PRVJT que funciona com um conjunto de soluções e permite soluções inviáveis que são penalizadas. Os pesos de penalização são controlados de forma adaptativa, dependendo se de viabilidade é fácil ou difícil de atingir. Em cada iteração exterior do algoritmo, as duas soluções A e B a partir da população são selecionados aleatoriamente. O algoritmo transforma A em B por uma série de 2-opt* e OR-opt movimentos. Isto gera uma série de soluções de entre as soluções A e B. No laço interno, algumas destas soluções são melhoradas usando vizinhos 2-opt*, intercâmbio, e Or-opt.

Certas soluções gerados são adicionados à solução de população se melhorar a qualidade da população, tendo em conta inviabilidade total inviabilidade tempo, e inviabilidade capacidade.

## 3.7 CONSIDERANDO OS MOTORISTAS, COM MÚLTIPLOS VEÍCULOS E MÚLTIPLAS JANELAS

O PRVJT com os regulamentos motorista leva em conta vários regulamentos referentes às horas de trabalho, pausas e descansos dos motoristas. Estes regulamentos estão aparecendo cada vez mais frequentemente em todo o mundo e podem variar de região para região. Para este problema, (GOEL; GRUHN, 2008) introduziram uma heurística LNS com base em operadores de pesquisa locais, (PRESCOTT-GAGNON; DESAULNIERS; ROUSSEAU, 2009) desenvolveram um método LNS com base na heurística de geração de colunas, e (KOK et al., 2010) propuseram uma heurística de programação dinâmica. De acordo com os resultados computacionais relatados obtidos em instâncias envolvendo 100 clientes ao longo de um horizonte de uma semana, o algoritmo de (PRESCOTT-GAGNON; DESAULNIERS; ROUSSEAU, 2009). supera os outros dois algoritmos.

O CVRP com o uso múltiplo de veículos é uma variante do CVRP clássica que permite atribuir várias rotas para o mesmo veículo ao longo de um horizonte de planejamento finito (por exemplo, o primeiro dia de trabalho) no contexto onde a disponibilidade do veículo, os custos de veículos fixo, ou tempo máximo de trabalho por veículo deve ser considerada. (AZI; GENDREAU; POTVIN, 2012) abordou o caso com janelas de tempo e projetado um método de solução de *Branch-and-Price* exata. Eles resolveram instâncias com até 40 clientes.

(DOERNER; SCHMID, 2010) estudaram a VRP com várias janelas de tempo interdependentes em que podem ser exigidas a cada cliente a ser visitado várias vezes e o tempo decorrido entre duas visitas consecutivas não deve exceder o tempo máximo. Essa variante do PRVJT é inspirada por um pedido de transporte de sangue. Os autores desenvolveram um algoritmo exato e algoritmos heurísticos. Os seus resultados mostram que os algoritmos heurísticos encontrar soluções razoavelmente perto de soluções ótimas numa fracção de um segundo.

# REFERÊNCIAS

AHUJA, R.; ORLIN, J.; SHARMA, D. Very large-scale neighborhood search. *International Transactions in Operational Research*, v. 7, n. 4-5, p. 301–317, set. 2000. ISSN 1475-3995. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1111/j.1475-3995.2000.tb00201.x/abstract>. Citado na página 43.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. *NBR 6028*: Resumo - apresentação. Rio de Janeiro, 2003. 2 p. Citado na página 11.

AZI, N.; GENDREAU, M.; POTVIN, J.-Y. A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research*, v. 199, n. 1, p. 103–112, out. 2012. ISSN 0254-5330, 1572-9338. Disponível em: <http://link.springer.com/article/10.1007/s10479-011-0991-3>. Citado 2 vezes nas páginas 37 e 53.

BALDACCI, R.; MINGOZZI, A.; ROBERTI, R. New Route Relaxation and Pricing Strategies for the Vehicle Routing Problem. *Oper. Res.*, v. 59, n. 5, p. 1269–1283, set. 2011. ISSN 0030-364X. Disponível em: <http://dx.doi.org/10.1287/opre.1110.0975>. Citado 2 vezes nas páginas 37 e 38.

BENAVENT, E. et al. A Branch-price-and-cut Algorithm for the Min-max K-vehicle Windy Rural Postman Problem. *Netw.*, v. 63, n. 1, p. 34–45, jan. 2014. ISSN 0028-3045. Disponível em: <http://dx.doi.org/10.1002/net.21520>. Citado 2 vezes nas páginas 37 e 38.

BENT, R.; HENTENRYCK, P. V. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, INFORMS, v. 38, n. 4, p. 515–530, 2004. Citado na página 48.

BRäYSY, O. *Local search and variable neighborhood search algorithms for the vehicle routing problem with time windows*. Tese (Doutorado) — University of Vaasa, Finland, 2001. Citado na página 40.

BRäYSY, O.; GENDREAU, M. Tabu Search heuristics for the Vehicle Routing Problem with Time Windows. *Top*, v. 10, n. 2, p. 211–237, dez. 2002. ISSN 1134-5764, 1863-8279. Disponível em: <http://link.springer.com/article/10.1007/BF02579017>. Citado na página 38.

BRäYSY, O.; GENDREAU, M. Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, v. 39, n. 1, p. 119–139, fev. 2005. ISSN 1526-5447. Disponível em: <http://dx.doi.org/10.1287/trsc.1030.0057>. Citado 3 vezes nas páginas 38, 39 e 43.

CATARINA A. S.; BACH, S. L. *Estudo do efeito dos parâmetros genéticos na solução otimizada e no tempo de convergência em algoritmos genéticos com codificações binária e real*. Dissertação (Mestrado) — Acta Scientiarum, Maringá, 2003. Citado 3 vezes nas páginas 21, 24 e 25.

COOK WILLIAM & RICH, J. A parallel cutting plane algorithm for the vehicle routing problem with time windows. technical report, computational and applied mathematics. 1999. Citado na página 38.

CORDEAU, J.-F.; LAPORTE, G.; MERCIER, A. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, Palgrave Macmillan UK, v. 52, n. 8, p. 928–936, 2001. Citado na página 46.

CORDEAU, J.-F.; LAPORTE, G.; MERCIER, A. Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, Palgrave Macmillan UK, v. 55, n. 5, p. 542–546, 2004. Citado na página 47.

CORDEAU, J.-F.; MAISCHBERGER, M. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, Elsevier, 2011. Citado na página 47.

CORDEAU, J.-F.; MAISCHBERGER, M. A Parallel Iterated Tabu Search Heuristic for Vehicle Routing Problems. *Comput. Oper. Res.*, v. 39, n. 9, p. 2033–2050, set. 2012. ISSN 0305-0548. Disponível em: <http://dx.doi.org/10.1016/j.cor.2011.09.021>. Citado na página 39.

Daniele Vigo; Paolo Toth (Ed.). *Vehicle Routing*. Society for Industrial and Applied Mathematics, 2014. (MOS-SIAM Series on Optimization). ISBN 978-1-61197-358-7. Disponível em: <http://epubs.siam.org/doi/book/10.1137/1.9781611973594>. Citado na página 45.

DELL?AMICO, M.; RIGHINI, G.; SALANI, M. A Branch-and-Price Approach to the Vehicle Routing Problem with Simultaneous Distribution and Collection. *Transportation Science*, v. 40, n. 2, p. 235–247, maio 2006. ISSN 0041-1655. Disponível em: <http://pubsonline.informs.org/doi/ref/10.1287/trsc.1050.0118>. Citado na página 36.

DESROCHERS, M. et al. Vehicle routing with time windows: Optimization and approximation. *Veh Rout Methods Stud*, v. 16, 01 1988. Citado na página 47.

DOERNER, K. F.; SCHMID, V. Survey: Matheuristics for Rich Vehicle Routing Problems. In: BLESA, M. J. et al. (Ed.). *Hybrid Metaheuristics*. Springer Berlin Heidelberg, 2010, (Lecture Notes in Computer Science, 6373). p. 206–221. ISBN 978-3-642-16053-0 978-3-642-16054-7. DOI: 10.1007/978-3-642-16054-7_15. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-16054-7_15>. Citado na página 53.

FALKENAUER, E.; BOUFFOUIX, S. A genetic algorithm for job shop. In: *Proceedings 1991 IEEE International Conference on Robotics and Automation*. Sacramento, CA: IEEE Computer Society Press, Los Alamitos, CA, 1991. v. 1, n. Cat. No. 91CH2969-4, p. 824–829. Citado na página 52.

GEHRING & Homberger benchmark. 2016. Disponível em: </projectweb/top/vrptw/homberger-benchmark/>. Citado na página 38.

GENDREAU, M.; LAPORTE, G.; POTVIN, J.-Y. The Vehicle Routing Problem. In: TOTH, P.; VIGO, D. (Ed.). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001. p. 129–154. ISBN 978-0-89871-498-2. Disponível em: <http://dl.acm.org/citation.cfm?id=505847.505853>. Citado na página 31.

GENDREAU, M.; POTVIN, J.-Y. *Handbook of Metaheuristics*. [S.l.]: Springer Science & Business Media, 2010. Google-Books-ID: xMTS5dyDhwMC. ISBN 978-1-4419-1665-5. Citado 2 vezes nas páginas 31 e 46.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997. ISBN 978-0-7923-9965-0. Citado na página 49.

GOEL, A.; GRUHN, V. A General Vehicle Routing Problem. *European Journal of Operational Research*, v. 191, n. 3, p. 650–660, dez. 2008. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221707001233>. Citado na página 53.

GOLDBERG, D. E.; ROTHLAUF, F. *Representations for Genetic and Evolutionary Algorithms: 104*. Heidelberg ; New York: Physica, 2002. ISBN 978-3-7908-1496-5. Citado 2 vezes nas páginas 19 e 22.

HASHIMOTO, H.; YAGIURA, M.; IBARAKI, T. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, v. 5, n. 2, p. 434–456, maio 2008. ISSN 1572-5286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1572528607000485>. Citado na página 47.

HASHIMOTO, H.; YAGIURA, M.; IBARAKI, T. An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization*, Elsevier, v. 5, n. 2, p. 434–456, 2008. Citado na página 52.

HILLIER, F. S.; LIEBERMAN, G. J. *Introduction to Operations Research and Revised CD-ROM 8*. [S.l.]: McGraw-Hill Science/Engineering/Math, 2005. ISBN 978-0-07-321114-5. Citado na página 38.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: University of Michigan Press, 1975. Google-Books-ID: JE5RAAAAMAAJ. ISBN 978-0-472-08460-9. Citado 2 vezes nas páginas 19 e 50.

JEPSEN, M. K.; SPOORENDONK, S.; RøPKE, S. A branch-and-cut algorithm for the symmetric two-echelon capacitated vehicle routing problem. *Transportation Science*, v. 47, p. 23–37, fev. 2013. ISSN 0041-1655. Citado 2 vezes nas páginas 37 e 38.

KALLEHAUGE, B.; LARSEN, J.; MADSEN, O. B. G. Lagrangian Duality Applied to the Vehicle Routing Problem with Time Windows. *Comput. Oper. Res.*, v. 33, n. 5, p. 1464–1487, maio 2006. ISSN 0305-0548. Disponível em: <http://dx.doi.org/10.1016/j.cor.2004.11.002>. Citado na página 38.

KINDERVATER, G. A.; SAVELSBERGH, M. W. Vehicle routing: handling edge exchanges. *Local search in combinatorial optimization*, Wiley Chichester, UK, p. 337–360, 1997. Citado na página 45.

KNIGHT, K. W.; HOFER, J. P. Vehicle Scheduling with Timed and Connected Calls: A Case Study. *Journal of the Operational Research Society*, v. 19, n. 3, p. 299–310, set. 1968. ISSN 0160-5682, 1476-9360. Disponível em: <http://link.springer.com/article/10.1057/jors.1968.73>. Citado na página 29.

KOK, A. L. et al. A dynamic programming heuristic for vehicle routing with time-dependent travel times and required breaks. *Flexible Services and Manufacturing Journal*, v. 22, n. 1-2, p. 83–108, jun. 2010. ISSN 1936-6582, 1936-6590. Disponível em: <http://link.springer.com/article/10.1007/s10696-011-9077-4>. Citado na página 53.

KOLEN, A. W. J.; KAN, A. H. G. R.; TRIENEKENS, H. W. J. M. Vehicle Routing with Time Windows. *Operations Research*, v. 35, n. 2, p. 266–273, abr. 1987. ISSN 0030-364X. Disponível em: <http://pubsonline.informs.org/doi/abs/10.1287/opre.35.2.266>. Citado na página 29.

KONDAGESKI, J. H. *Ajuste de Modelos de Qualidade da Água para Rio Algoritmo Genético*. Dissertação (Mestrado) — Universidade Federal do Paraná, Curitiba, 2008. Citado na página 20.

LAND A. H., A. G. D. An automatic method of solving discreteprogramming problems. *Econometrica*, v. 28, p. 497?–520, 1960. Citado na página 35.

LAPORTE, G. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, v. 59, n. 3, p. 345–358, jun. 1992. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/037722179290192C>. Citado na página 27.

LAPORTE, G.; TOTH, P.; VIGO, D. Vehicle routing: historical perspective and recent contributions. *EURO Journal on Transportation and Logistics*, v. 2, n. 1-2, p. 1–4, maio 2013. ISSN 2192-4376, 2192-4384. Disponível em: <http://link.springer.com/article/10.1007/s13676-013-0020-6>. Citado 2 vezes nas páginas 28 e 29.

LARSEN, A. *The Dynamic Vehicle Routing Problem*. Tese — Department of Mathematical Modelling, The Technical University of Denmark, Building 321, DTU, DK-2800 Kgs. Lyngby, 2000. Disponível em: <http://www2.imm.dtu.dk/pubdb/p.php?143>. Citado na página 38.

LIEBERMAN, F. S. H. . M. G. J. *Operations Research An Introduction*. Beijing: Tsinghua University Press. Pub. Date :2010-03, 2010. ISBN 978-7-302-21565-3. Citado 2 vezes nas páginas 27 e 35.

LINDEN, R. *Algoritmos Genéticos*. Edição: 3ª. [S.l.]: Ciência Moderna, 2012. ISBN 978-85-399-0195-1. Citado 2 vezes nas páginas 19 e 21.

MOGNON, V. R. *Algoritmos Genéticos Aplicados na Otimização de Antenas*. Dissertação (Mestrado) — Universidade Federal do Paraná, Curitiba, 2004. Citado 3 vezes nas páginas 21, 23 e 24.

NAGATA, Y. New Approach of a Genetic Algorithm for TSP Using the Evaluation Function Considering Local Diversity Loss. *Transactions of the Japanese Society for Artificial Intelligence*, v. 21, p. 195–204, 2006. Citado na página 52.

NAGATA, Y.; BRÄYSY, O.; DULLAERT, W. A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, Elsevier, v. 37, n. 4, p. 724–737, 2010. Citado 3 vezes nas páginas 45, 46 e 47.

PRESCOTT-GAGNON, E.; DESAULNIERS, G.; ROUSSEAU, L.-M. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, Wiley Online Library, v. 54, n. 4, p. 190–204, 2009. Citado na página 53.

PULLEN, H. G. M.; WEBB, M. H. J. A computer application to a transport scheduling problem. *The Computer Journal*, v. 10, n. 1, p. 10–13, jan. 1967. ISSN 0010-4620, 1460-2067. Disponível em: <http://comjnl.oxfordjournals.org/content/10/1/10>. Citado na página 29.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach: United States Edition*. Edição: 1. Englewood Cliffs, N.J: Pearson, 1994. ISBN 978-0-13-103805-9. Citado na página 39.

SAVELSBERGH, M. W.; SOL, M. The general pickup and delivery problem. *Transportation science*, [Baltimore]: Transporation Science Section of ORSA, 1967-, v. 29, n. 1, p. 17–29, 1995. Citado na página 29.

SHAW, P. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In: MAHER, M.; PUGET, J.-F. (Ed.). *Principles and Practice of Constraint Programming ? CP98*. Springer Berlin Heidelberg, 1998, (Lecture Notes in Computer Science, 1520). p. 417–431. ISBN 978-3-540-65224-3 978-3-540-49481-2. DOI: 10.1007/3-540-49481-2_30. Disponível em: <http://link.springer.com/chapter/10.1007/3-540-49481-2_30>. Citado na página 48.

SOLOMON, M. M. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Oper. Res.*, v. 35, n. 2, p. 254–265, apr 1987. ISSN 0030-364X. Disponível em: <http://dx.doi.org/10.1287/opre.35.2.254>. Citado 4 vezes nas páginas 29, 33, 36 e 41.

TAILLARD, É. D. Comparison of non-deterministic iterative methods. In: *Metaheuristic Interantional Conference (MIC'01) proceedings*. [s.n.], 2001. Disponível em: <http://mistic.heig-vd.ch/taillard/articles.dir/Taillard2001.pdf>. Citado na página 40.

UNO, T.; YAGIURA, M. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, Springer, v. 26, n. 2, p. 290–309, 2000. Citado na página 45.

YVES, J. F.-X. C.; LABURTHE, L. F. Combining sets, search, and rules to better express algorithms. *ICLP99*, 1999. Citado na página 40.

# APÊNDICES

# APÊNDICE A – QUISQUE LIBERO JUSTO

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

# ANEXOS

# ANEXO A – MORBI ULTRICES RUTRUM LOREM.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.