UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Daniel Vieira de Souza

UMA ABORDAGEM POR HIPERHEURÍSTICA COM APRENDIZADO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO

Mossoró - RN
2019

Daniel Vieira de Souza

UMA ABORDAGEM POR HIPERHEURÍSTICA COM APRENDIZADO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Profº Dr. Francisco Chagas de Lima Júnior
Coorientador: Profº Dr. Carlos Heitor Pereira Liberalino

Mossoró - RN
2019

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

S729a Souza, Daniel Vieira de Souza
UMA ABORDAGEM POR HIPERHEURÍSTICA COM APRENDIZADO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO. / Daniel Vieira de Souza Souza. - Mossoró, 2019.
78p.

Orientador(a): Prof. Dr. Francisco Chagas de Lima Júnior Lima Júnior.
Coorientador(a): Prof. Dr. Carlos Heitor Pereira Liberalino Liberalino.
Dissertação (Mestrado em Programa de Pós-Graduação em Ciência da Computação). Universidade do Estado do Rio Grande do Norte.

1. Hiperheurística com aprendizado. 2. Metaheurística GRASP. 3. Problema de Roteamento de Veículos com Janela de Tempo. 4. Aprendizado por Reforço. I. Lima Júnior, Francisco Chagas de Lima Júnior. II. Universidade do Estado do Rio Grande do Norte. III. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pela Diretoria de Informatização (DINF), sob orientação dos bibliotecários do SIB-UERN, para ser adaptado às necessidades da comunidade acadêmica UERN.

DANIEL VIEIRA DE SOUZA

UMA ABORDAGEM POR HIPERHEURISTICA COM APRENDIZADO PARA O PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO.

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 22 / 03 / 2019

Prof. Dr. FRANCISCO CHAGAS DE LIMA JÚNIOR
Orientador e Presidente

Prof. Dr. CARLOS HEITOR PEREIRA LIBERALINO
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. FRANCISCO DANTAS DE MEDEIROS NETO
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. ANDRE PEDRO FERNANDES NETO
Universidade Federal Rural do Semi-Árido - UFERSA

Prof. Dr. GUSTAVO AUGUSTO LIMA DE CAMPOS
Universidade Estadual do Ceará - UECE

Dedico este trabalho a DEUS, essencial na minha vida. Também a toda minha família que sempre esteve presente me dando o incentivo e apoio que eu precisava.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, que iluminou meu caminho até aqui, me dando coragem e motivos para seguir em frente mesmo com os obstáculos encontrados.

Agradeço a minha família pelo apoio e amor incondicional, sempre me apoiando nos momentos difíceis e de incerteza, me encorajando e compreendendo nas situações mais adversas e principalmente me fazendo acreditar que eu sempre seria capaz.

Agradeço aos professores Lima Júnior e Heitor Liberalino pela orientação no trabalho, paciência e compreensão que sempre tiveram, sempre dispostos a ajudar.

A UERN e UFERSA por fornecer o ambiente necessário para a realização deste trabalho. Aos professores do mestrado, dos quais concretizei muitas amizades, pelo conhecimento e experiências compartilhadas.

À CAPES, pelo apoio financeiro que viabilizou a realização deste trabalho

Agradeço aos todos os colegas de sala pela ajuda constante durante essa caminhada, em especial a meus amigos Felipe Rodrigues, Erico Gomes e Wedson Carlos.

"E a paz de Deus, que excede todo entendimento, guardará os vossos corações e os vossos pensamentos em Cristo Jesus."
Filipense 4:7

RESUMO

O conceito de hiperheurística é um tanto novo no ramo da otimização, se trata de um método que propõe uma estratégia de resolução que opera em um novo nível de abstração, onde sem a utilização de informações específicas do problema tratado o método é capaz de oferecer soluções através do gerenciamento de um conjunto de métodos heurísticos disponíveis, podendo ainda serem empregados mecanismos de aprendizado e/ou treinamento. Essas características permitem que esse tipo de abordagem possa se adaptar a diversos domínios de problemas ou a diferentes classes de instâncias. Principalmente em problemas onde dispondo de um conjunto de métodos heurísticos não se sabe que técnica de resolução é a mais adequada. O presente trabalho propõe como abordagem uma hiperheurística com aprendizado integrada à Metaheurística *GRASP (Greedy Randomized Adaptive Search Procedure)* aplicada à uma variante do clássico Problema de Roteamento de Veículos (PRV), o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT). Tendo esse método como mecanismo de aprendizado uma técnica de Aprendizado por Reforço (AR), o algoritmo *Q-Learning*, que terá a tarefa de indicar que método heurístico mais adequado irá compor a fase construtiva do *GRASP*. Assim como a hiperheurística com aprendizado foi implementado um outro método hiperheurístico de gerenciamento de heurísticas aleatório, buscando comparar e analisar o desempenho do método de aprendizado. Os algoritmos foram testados em experimentos computacionais com instâncias conhecidas da literatura para o PRVJT e os resultados obtidos comparados ao custo de solução, tempo de execução e escolha do método heurístico construtivo utilizado na fase de construção do *GRASP*.

Palavras-chave: Hiperheurística com aprendizado, Metaheurística *GRASP*, Problema de Roteamento de Veículos com Janela de Tempo, Aprendizado por Reforço.

ABSTRACT

The concept of hyperheuristics is somewhat new in the field of optimization, it is a method that proposes a strategy of resolution that operates in a new level of abstraction, where without the use of specific information of the treated problem the method is able to offer solutions through the management of a set of available heuristic methods, and learning and / or training mechanisms may be employed. These characteristics allow this type of approach to adapt to different problem domains or to different classes of instances. Especially in problems where having a set of heuristic methods is not known which technique of resolution is the most adequate. The present work proposes as approach a hyperheuristic with learning integrated to *GRASP (Greedy Randomized Adaptive Search Procedure)* Metaheuristic applied to a variant of the classic Vehicle Routing Problem (PRV), the Vehicle Routing with Time Window (PRVJT). Having this method as learning mechanism a Reinforcement Learning (RA) technique, the algorithm *Q-Learning* that will have the task of indicating which heuristic method is most suitable to compose the constructive phase of *GRASP*. Like hyperheuristics with learning, another hyperheuristic method of managing random heuristics was implemented, trying to compare and analyze the performance of the learning method. The algorithms were tested in computational experiments with known instances in the literature for the PRVJT and the results obtained compared to the cost of solution, execution time and choice of the constructive heuristic method used in the *GRASP* construction phase.

Key-words: Hyperheuristics with Learning, GRASP Metaheuristics, Vehicle Routing Problem with Time Window, Reinforcement Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
AR	Aprendizagem por Reforço
BCRC	Best Cost Route Crossover
CRRM	Constrained Route Reversal Mutation
CV	Clark & Wrigth
GRASP	Greedy Randomized Adaptive Search Procedure
LC	Lista de Candidatos
LRC	Lista Restrita de Candidatos
MACS-VRPTW	Multiple Ant Colony System for Vehicle Routing Problem with Time Windows
NN	Nearest Neighbor
PCV	Problema do Caixeiro Viajante
PDM	Processo de Decisão de Markov
PFIH	Push Forward Insertion Heuristic
PRV	Problema de Roteamento de Veículos
PRVB	Problema de Roteamento de Veículos com Backhauls
PRVC	Problema de Roteamento de Veículos Capacitado
PRVCE	Problema de Roteamento de Veículos Coleta e Entrega
PRVE	Problema de Roteamento de Veículos Estocástico
PRVEP	Problema de Roteamento de Veículos com Entrega Particionada
PRVJT	Problema de Roteamento de Veículos com Janela de Tempo
PRVLD	Problema de Roteamento de Veículos com Limite de Distância
PRVMD	Problema de Roteamento de Veículos com Múltiplos Depósitos
PRVP	Problema de Roteamento de Veículos Periódico

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Problemas de otimização combinatória são aqueles que envolvem um conjunto finito de possíveis soluções, em que se almeja maximizar ou minimizar uma função objetivo. A maneira mais adequada de se resolver um problema desse tipo é analisando todo o espaço de busca, procurando pela melhor solução existente, ou seja, aquela que ofereça o melhor valor para a função objetivo. Porém isso nem sempre é possível, já que há problemas em que o espaço de busca por soluções é muito grande, o que torna tal abordagem impraticável, mesmo que seja executada em um equipamento de última geração. Tais problemas muitas vezes representam situações reais vividas no cotidiano, o que motiva pesquisadores da área a se concentrarem em formulá-los matematicamente e resolvê-los de modo possível e eficiente (ALAZZAM; LEWIS III, 2013) (TANDABANI et al., 2016).

Muitos desses problemas são pertencentes a classe dos problemas NP-difíceis, o que significa dizer que não se conhece ainda um algoritmo que os resolvam de forma exata em tempo polinomial, considerando instâncias de grande porte. Diante dessa impossibilidade de se fazer o uso de um método exato como resolução, pesquisadores têm trabalhado no desenvolvimento de métodos não-exatos, no objetivo de encontrar uma solução ótima, ou ainda uma solução aproximada com valores aceitáveis em um tempo computacional viável. Um tipo mais sofisticado desses métodos, denominado metaheurística surgiu na década de 1980 e pode ser definido como: algoritmos aproximativos que utilizam de processos iterativos para guiar heurísticas subordinadas por meio de uma combinação inteligente de diferentes conceitos, visando uma exploração/explotação do espaço de busca, a fim de encontrar, de forma eficiente, soluções de alta qualidade (BLUM; ROLI, 2003).

Como exemplo de metaheurísticas podemos citar: Algoritmos Genéticos (HOLLAND, 1975), *Simulated Annealing* (KIRKPATRICK et al., 1983), Busca Tabu (GLOVER, 1986), *Greedy Randomized Adaptive Search Procedure - GRASP* (FEO; RESENDE, 1995), Otimização por Colônia de Formigas (DORIGO, 1999).

A estratégia de alto nível das metaheurísticas possibilitou uma abstração do domínio de problema não presente nas heurísticas, o que permitiu sua aplicação em vários tipos de problema. Contudo a implementação de uma metaheurística faz o uso de informações específicas do problema requerendo um conhecimento prévio do mesmo. Também é importante destacar que para que esse tipo de método tenha um bom desempenho há a necessidade de ajustes de seus parâmetros, o que influência

diretamente na qualidade das soluções geradas e no tempo de execução dos algoritmos, podendo assim haver métodos ou ajustes que sejam mais indicados à um dado problema ou classe de instância específica (CHAKHLEVITCH; COWLING, 2008).

Apesar do sucesso com as metaheurísticas na implementação de vários problemas de otimização, segundo Burke et al., (2003) isso não foi refletido na adoção dessas técnicas em empresas do comércio e da indústria. Os autores atribuem isso ao motivo dos métodos estarem muito ligados ao domínio do problema ou necessitarem de um alto grau de conhecimento de suas parametrizações. Foi discutido ainda que pequenas empresas não precisam necessariamente que seus problemas sejam resolvidos de forma ótima ou ainda de forma aproximada. Essas organizações têm interesse em soluções de boa qualidade e rápidas.

Diante disso surgiu como uma alternativa de resposta o conceito de hiperheurística definido por Cowling et al. (2001) como: uma heurística que lida indiretamente com o problema a ser resolvido, por meio do gerenciamento de heurísticas. Usando esse conceito um método pode se adaptar a qualquer problema de otimização, necessitando apenas de: um conjunto de heurística básicas que possam obter uma solução para um problema, uma solução inicial e uma função que possa analisar o desempenho dessas heurísticas avaliando o custo das soluções.

O presente trabalho propõe uma estratégia hiperheurística como solução para uma variante bem conhecida do clássico problema de otimização Problema de Roteamento de Veículos (PRV), o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), o qual é amplamente estudado na literatura por ser comparado a vários problemas reais inseridos no cotidiano de organizações, seja no setor comercial, industrial ou de serviços. Devido sua natureza combinatória, esse tipo de problema, como já dito, não pode ser tratado eficiente por métodos exatos pelo alto grau de combinações em possíveis soluções, não sabendo ainda da existência de um algoritmo exato que possa resolvê-los em tempo viável. O que torna o uso de estratégias heurísticas como resolução uma opção praticável.

O método proposto como solução para esse problema é uma estratégia hiperheurística online com aprendizagem integrado a metaheurística *GRASP*. Nessa abordagem é usado como mecanismo de aprendizado uma técnica de Aprendizado por Reforço (AR), o algoritmo *Q-Learning*, com o objetivo de decidir dentre um conjunto de heurísticas de construção, qual método mais adequado deve ser utilizado na fase construtiva do *GRASP*.

1.2 MOTIVAÇÃO

O trabalho propõe uma nova abordagem para o PRVJT, através de uma estratégia que esteja melhor adaptada ao domínio do problema ou ainda a uma classe de instância

específica. Na metaheurística *GRASP* podemos perceber uma variação da qualidade das soluções em suas parametrizações, mais especificamente no parâmetros α, que pode ter valores diferentes em distintas classes de instâncias de um problema.

Essa variação também pode acontecer na escolha dos métodos que compõem suas fases de construção e de busca local, pois a estrutura do método *GRASP* permite várias implementações de métodos construtivos e de busca local diferentes, já que suas fases são independentes. Com a variedade de métodos heurísticos presentes na literatura, não se pode prever que método ou ajuste deste pode oferecer uma melhor solução inicial ou ainda um mecanismo de busca local mais eficiente. Apenas através da aplicação e implementação de tais técnicas ao problema tratado pode-se avaliar o desempenho que cada configuração pode ter.

Diante da quantidade de métodos heurísticos a serem aplicados ao PRVJT, tanto para gerar uma solução inicial como para serem utilizados em uma busca local surgiu a ideia de desenvolver uma estratégia que esteja melhor ajustada para resolver uma instância específica do problema. Essa estratégia utiliza o conceito de hiperheurística com aprendizado para gerenciar um conjunto de heurísticas disponíveis para o problema, visando se adequar a um problema específico utilizando o método que seja mais eficiente para a fase construtiva da metaheurística *GRASP*. Pois é nessa fase que são geradas as primeiras soluções, as quais podem oferecer ótimos locais promissores para a sequência do método.

Para decidir que heurística construtiva ou ainda quais seus parâmetros usar na fase construtiva do *GRASP*, a hiperheurística terá como aprendizado o uso de uma técnica de Aprendizagem por Reforço (AR), na qual o algoritmo *Q-Learning* terá como tarefa indicar qual método seria mais adequado.

1.3 OBJETIVO GERAL

O presente trabalho tem como objetivo a resolução do Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) através de uma abordagem por Hiperheurística com aprendizado integrada a metaheurística *GRASP*. Utilizando técnica de AR, o algoritmo *Q-Learning* como aprendizado, o qual terá a tarefa de sugerir dentro de um conjunto de métodos heurísticos construtivos qual seria o mais adequado para fazer parte da fase de construção do *GRASP*.

1.4 OBJETIVOS ESPECÍFICOS

Como objetivos específicos temos:

- Modelar o problema de aprendizado como um PDM;

- Desenvolver a estratégia que irá indicar qual heurística construtiva deve ser utilizada na fase de construção da metaheurística *GRASP*, utilizando o algoritmo *Q-learning*;
- Implementar a Hiperheurística construtiva aleatória integrada a metaheurística *GRASP*, aplicada ao PRVJT;
- Implementar a Hiperheurística online com aprendizado integrada a metaheurística *GRASP*, aplicada ao PRVJT;
- Realizar experimentos computacionais utilizando instâncias consagradas na literatura;
- Comparar os resultados obtidos pelos métodos propostos

1.5 ORGANIZAÇÃO DO TEXTO

Este trabalho encontra-se dividido em seis capítulos. No capítulo 2 é apresentado um referencial teórico do trabalho abordando sobre o Problema de Roteamento de Veículos (PRV), sua variante, o Problema de Roteamento de Veículos (PRVJT) que será o problema tratado, a técnica de Aprendizado por Reforço (AR) a ser utilizada pelo método proposto, o algoritmo *Q-learning* e os conceitos sobre o tipo de abordagem aplicada ao problema considerado, as hiperheurísticas. No capítulos 3 são comentados os trabalhos relacionados à abordagens para o PRVJT, além de trabalhos sobre métodos hiperheurísticos disponíveis na literatura. No capítulo 4 o método proposto e a modelagem para o problema de aprendizado são descritas. No capítulo 5 as instâncias de testes, os experimentos computacionais e resultados obtidos são apresentados e discutidos. No capítulo 6 as conclusões e sugestões de trabalho futuros são mostrados.

2 REFERENCIAL TEÓRICO

2.1 PROBLEMAS DE ROTEAMENTO E APLICAÇÕES

Segundo Goldbarg e Luna (2000) um sistema de roteamento pode ser entendido como um conjunto organizado de meios que têm por objetivo o atendimento de demandas posicionadas em arcos ou em vértices de uma rede de transportes. Podendo esse estar dividido em três setores: estratégico, tático e lógico.

As decisões do setor estratégico estão voltadas a aspectos que sofrem influências externas e determinam sua organização, com impacto sobre todo o sistema e de efeito prolongado. Destacando ainda que tais decisões tomadas de maneira errada acarretarão em dificuldades maiores na operação e otimização do sistema. No setor tático as decisões são tomadas de maneira mais específicas, tendo em vista a construção do sistema. Finalmente no setor logístico as decisões sobre os aspectos mais versáteis de todo processo (GOLDBARG; LUNA, 2000).

O objetivo maior da logística é fazer chegar provisões e/ou serviços a pontos de consumo, a partir de pontos de suprimento (BODIN et al., 1983), executando um plano econômico e flexível para atender a rede de demandas. Nas atividades do setor de logística que os Problemas de Roteamento de Veículos (PRV) estão inseridos. Nesses problemas as decisões envolvem características combinatória e de solução complexa. Como situações envolvidas nos PRV tradicionais podemos citar: clientes servidos por depósitos, demanda de clientes, tipo de veículo utilizado (capacidade, velocidade), etc.

Diante das situações apresentadas se objetiva o planejamento de um roteamento de veículos e curso de atividades que orientem à minimização de custos, sejam esses: prazos de entrega, distância percorrida pelos veículos, caminhos a percorrer (combustível, manutenção, tempo de operação), alocação de mão-de-obra, número de veículos empregados, etc (GOLDBARG, 2005).

Inserido no problema de roteamento de veículos pode-se ressaltar o subproblema de distribuição, o qual envolve custos bem elevados. Hollander (1987, apud GOLDBARG, 2005) indica que o atraso na entrega de produtos no comércio colabora com um custo final do produto acrescido de 5% por mês. Bodin et al. (1983) mostra que a distribuição física dos produtos contribui com aproximadamente 16% do custo final da mercadoria.

Das atividades da cadeia logística o transporte é aquele que detém a maior parte dos custos, atingindo de um a dois terços do valor total. Portanto reduzir esses custos significa diminuir o preço do produto final, além de aumentar o lucro. De outro modo, algumas mercadorias necessitam de um transporte não apenas econômico, mas também seguro, como é o caso da distribuição de combustíveis e medicamentos (BRÄYSY;

GENDREAU, 2005).

Esse problema dispõe de uma quantidade muito vasta de aplicações práticas, já que essas fazem parte de situações vividas principalmente no cotidiano da indústria, comércio e setor de serviços.

2.2 PROBLEMA DE ROTEAMENTO DE VEÍCULOS (PRV)

Segundo Dantzig e Ramser (1959), o Problema de Roteamento de Veículos (PRV) é considerado uma extensão do conhecido Problema do Caixeiro Viajante (PCV), o qual na sua versão mais tradicional, procura identificar a rota de menor custo em que todos os clientes sejam atendidos, partindo de um ponto inicial e retornando a esse. No PRV a demanda de um conjunto de clientes deve ser atendida por uma frota de veículos, de modo que o menor custo seja alcançado. A diferença entre o PRV e o PCV como se pode notar está no número de rotas obtidas, no caso do PCV apenas uma, já o PRV depende do número de veículos empregados.

No PRV devemos realizar a distribuição de bens (ou serviços) de um depósito central para usuários finais (clientes). Essa tarefa é feita por um conjunto de veículos que partem do depósito atendendo as demandas de todo o conjunto de clientes geograficamente distribuídos até retornarem ao depósito finalizando a rota. O objetivo é determinar as rotas que realizem tal atendimento com o menor custo, seja esse a distância mínima que os veículos devem percorrer e/ou número mínimo de veículos utilizados.

Pertencente a uma categoria de problemas mais ampla da Pesquisa Operacional denominada problemas de otimização em rede, o PRV é um problema de programação inteira que pertence a categoria dos problemas de complexidade NP-difíceis. Portanto, conforme o tamanho do problema, ainda não se sabe de um algoritmo capaz de obter a solução ótima do problema em tempo polinomial. Dessa maneira, o uso de métodos heurísticos e meta-heurísticos são uma alternativa, já que esses possuem capacidade de alcançar soluções aproximadas a solução ótima, consumindo tempo e esforços computacionais relativamente pequenos, quando comparados aos custos empregados nos métodos exatos. A seguir a modelagem matemática que descreve formalmente esse problema.

2.2.1 Modelo Matemático para o PRV

Segundo Toth e Vigo (2002) o PRV pode ser modelado como um grafo completo e não direcionado por $G = (V, A)$, formado por um conjunto V de N vértices e um conjunto A de arestas, sendo $V = \{0, 1, 2, \ldots N\}$ e $A = \{(i, j) \mid i, j \in V, i \neq j\}$, respectivamente. Neste caso, o conjunto V possui N vértices, que representa os clientes e o depósito, onde a

quantidade de clientes a serem atendidos é dada por $N - 1$ e cada cliente i possui uma demanda q_i. O índice i do depósito é representado por 0 e esse tem $q_i = 0$. O conjunto A de arestas representa as ligações entre os vértices, em que cada uma das arestas (i, j) tem um custo de viagem c_{ij} associado, equivalente à distância entre dois clientes i e j. O número de veículos disponíveis é igual a $|K|$, onde cada veículo tem um índice k que vai de 1 a $|K|$. A frota de veículos é homogênea, ou seja, todos os veículos tem capacidade igual a C. Também são utilizadas no modelo duas variáveis de decisão: a variável x_{ijk}, a qual tem seu valor igual a 1 se um veículo parte de um cliente i para um cliente j e valor igual 0 caso contrário. A variável y_{ik} que tem valor igual 1 quando um cliente i é atendido por um veículo k. Com o modelo busca-se minimizar o custo total das distâncias percorridas pelas rotas estabelecidas para cada veículo. O modelo é dado por:

$$min \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^{K} x_{ijk} \tag{2.1}$$

Sujeito a:

$$\sum_{k=1}^{K} y_{ik} = 1 \quad \forall i \in V \backslash \{0\} \tag{2.2}$$

$$\sum_{k=1}^{K} y_{0k} = K \tag{2.3}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{ijk} = y_{ik}, \forall i \in V, k = 1, \ldots, K \tag{2.4}$$

$$\sum_{i \in V} q_i y_{ik} \leq C, \forall k = 1, \ldots, K \tag{2.5}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, \forall S \subseteq V \backslash \{0\}, |S| \geq 2, k = 1, \ldots, K \tag{2.6}$$

$$x_{ijk} \in \{0, 1\}, \forall i, j \in V, \forall k = 1, \ldots, K \tag{2.7}$$

$$y_{ik} \in \{0, 1\}, \forall i \in V, \forall k = 1, \ldots, K \tag{2.8}$$

A função objetivo (2.1) do modelo matemático acima têm como fim minimizar os custos das viagens feitas pelas rotas de todos os veículos. As restrições (2.2), (2.3) e (2.4) respectivamente obrigam que cada cliente seja visitado uma única vez, que o número de veículos que partem do depósito seja igual a $|K|$ e que um dado veículo entre e saia de um cliente i. A restrição (2.5) assegura que o total de demandas atendidas por um veículo não exceda sua capacidade. A restrição (2.6) garante a eliminação de sub rotas que não incluam o depósito. As restrições (2.7) e (2.8) definem valores binários as variáveis de decisão x_{ijk} e y_{ik}.

2.2.2 Variantes do PRV

Mesmo o PRV sendo um problema complexo, esse ainda não pode expressar algumas das situações presentes no cotidiano da área logística. Sendo preciso que certas restrições sejam acrescentadas, para que serviços relacionados com clientes, veículos ou depósitos sejam então representados atendendo as características reais do problema. O PRV está entre os mais complexos da área de otimização combinatória. Pela numerosa quantidade de variáveis, variedade de restrições e objetivos contidos. É necessária uma análise cautelosa de suas possíveis características para uma melhor compreensão. Uma categorização clássica é apresentada em Bodin e Golden (1981) que classifica o PRV segundo os critérios:

- Tempo para servir um cliente
  - Tempo fixo,
  - Janela de Tempo.
- Número de depósitos:
  - Um,
  - Mais de um.
- Tamanho da frota de veículos:
  - Um veículo,
  - Mais de um veículo.
- Tipo de frota disponível:
  - Homogênea,
  - Heterogênea.
- Natureza da demanda e parâmetros:
  - Determinística,
  - Estocástica.
- Localização da demanda:
  - Nos vértices,
  - Nos arcos.
- Grafo de substrato:
  - Direcionado,
  - Não direcionado,
  - Misto.

- Restrições na capacidade de veículos:
  - Sujeitos às mesmas restrições,
  - Restrições diferentes.
- Tempo de roteamento:
  - O mesmo para todos os veículos,
  - Tempos diversos,
  - Sem restrições de tempo.
- Custos:
  - Variáveis (associados à rota escolhida),
  - Fixos.
- Operação:
  - De entrega,
  - De recolhimento,
  - Ambas.
- Objetivo:
  - Minimizar custos fixos,
  - Minimizar custos de operação na rota,
  - Minimizar o número de veículos.
- Restrições na capacidade dos arcos:
  - Imposta a todos os arcos,
  - Impostas a um subconjunto de arcos,
  - Sem restrições.

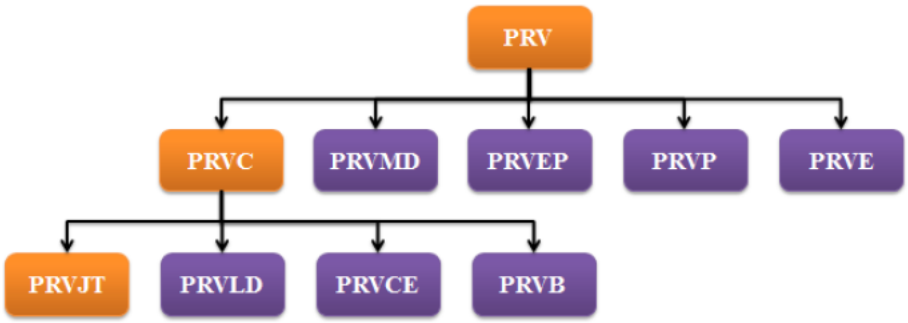
Consequentemente a cada modificação do PRV original eleva-se a complexidade do modelo, onde o acréscimo de novas restrições é necessário para representar mais fielmente as características do problema. Existem inúmeras variações disponíveis na literatura dentre as mais conhecidas podemos citar:

- PRV Capacitado (PRVC): são impostas apenas restrições quanto a capacidade dos veículos utilizados. Sendo necessário verificar se a soma das demandas dos clientes designados a rota de um veículo não ultrapassa sua capacidade.

- PRV com Janela de Tempo (PRVJT): cada cliente deve ser atendido dentro de um intervalo de tempo (janela de tempo). O veículo deve iniciar o serviço com o cliente dentro da janela de tempo e permanecer até que o atendimento termine. Caso o veículo chegue antes do início da janela de tempo, ele deve esperar o começo;
- PRV com múltiplos Depósitos (PRVMD): as rotas dos veículos podem ter início e fim em qualquer um dos depósitos existentes;
- PRV com Entrega Particionada (PRVEP): um único cliente pode ser incluído em diferentes rotas e atendido por vários veículos;
- PRV Periódico (PRVP): o atendimento de todos os clientes é feito em um período de dias;
- PRV Estocástico (PRVE): pelo menos uma variável do problema deve ser aleatória;
- PRV com Limite de Distância (PRVLD): os trajetos de cada veículo não devem ultrapassar um limite máximo de distância;
- PRV com Coleta e Entrega (PRVCE): há clientes com demanda de serviços de entrega e coleta em uma rota; e
- PRV com Backhauls (PRVB): a palavra Backhauls se refere a clientes que possuem uma demanda a ser enviada para o depósito.

A Figura 1 exemplifica as variantes do PRV.

Figura 1 – Variantes do PRV (STEHLING, 2015).



O presente trabalho irá abordar a variante do PRV conhecida como Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) descrito em mais detalhes a seguir.

2.3 PROBLEMA DE ROTEAMENTO DE VEÍCULOS COM JANELA DE TEMPO (PRVJT)

O Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) é uma extensão do PRV original, o qual adiciona ao problema clássico um limite de horário para atender as demandas dos clientes. Essa restrição é conhecida como janela de tempo, com a qual pode-se representar melhor situações em que o horário de atendimento é um aspecto importante, trazendo essa representação mais próxima as atividades executadas no cotidiano de empresas da logística e distribuição de bens e serviço.

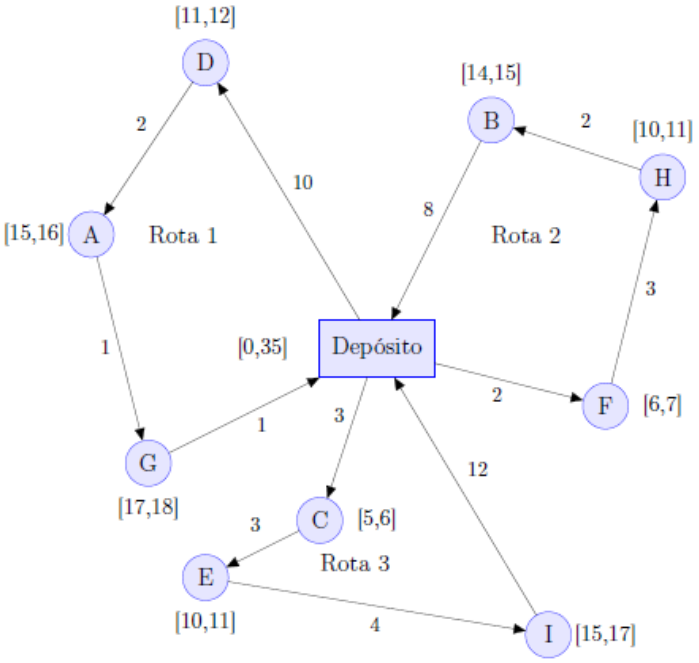
Em um cenário real essas empresas possuem horários de início e fim de operação e jornadas de trabalho a considerar, além de restrições de horário em que portos operam ou ainda relacionadas a circulação de alguns tipos de veículo em cidades e vias. Segundo Gendreau e Tarantilis (2010), o PRVJT consiste em um problema cujo o objetivo é minimizar o custo total das rotas estabelecidas entre o depósito e os clientes distribuídos geograficamente. Onde um conjunto de M veículos partem do depósito visando atender as demandas de um conjunto de N cliente, respeitando as restrições impostas pela janela do depósito e de cada cliente. Conforme Cordeau et al. (2001) esse objetivo pode ser decomposto em duas parcelas: minimizar o número total de veículos necessários para gerar as rotas e minimizar a distância total percorrida nas rotas por esses veículos.

O PRVJT pode ser resumido a quatro componentes: depósito, clientes, veículos e rotas. As características inerentes a cada componente do problema são:

- Depósito único;
- A capacidade dos veículos é conhecida;
- A frota de veículos disponíveis para o atendimento é homogênea;
- Todos os veículos devem permanecer no local do atendimento até que esse serviço acabe;
- Cada rota é representada por apenas um veículo e cumprida apenas por ele;
- Toda rota gerada deve ter como ponto de partida e chegada o depósito;
- Existe um custo correspondente ao deslocamento do veículos que atende um cliente i para um cliente j. A soma dos deslocamentos dos veículos para atender os clientes designados a uma rota é a distância percorrida por aquela rota;
- Cada cliente possui uma demanda conhecida;
- Cada cliente i possui um tempo de serviço s_i para que o atendimento seja realizado;
- Um cliente deve ser atendido por apenas um veículo;

- A soma das demandas dos clientes atendidos por um veículo não deve exceder sua capacidade;
- Os veículos realizam todos os atendimentos dentro de um horário de funcionamento, caracterizando uma janela de tempo $[T_i, T_f]$ para o depósito, em que os atendimentos devem ser feitos num período entre o início $[T_i]$ e o fim $[T_f]$ do horário de funcionamento;
- Cada cliente possui uma janela de tempo, característica essa que dita o horário máximo e mínimo em que um cliente pode ser atendido;
- Caso algum veículo designado para realizar um serviço a um cliente chegue depois do horário limite para o atendimento, esse não poderá ser realizado; e
- Caso um veículo designado para realizar um serviço a um cliente chegue antes do horário de início que o atendimento pode ser feito, esse deve esperar o tempo suficiente até a o instante de tempo igual ao horário de início da janela de tempo do cliente seja alcançado.

Figura 2 – Exemplo de solução para o PRVJT (LIMA, 2013).



A Figura 2 ilustra uma solução para o PRVJT. Nela o depósito é representado pelo índice 0. As rotas formadas são três R_1, R_2, R_3, onde cada uma abriga seus respectivos clientes na ordem com que são atendidos, além da distância percorrida. $R_1 = \{0 - F - H - B - 0\}$ com 14 unidades, $R_2 = \{0 - D - A - G - 0\}$ com 15 unidades, $R_3 = \{0 - C - E - I - 0\}$ com 16 unidades. Dada a ordem de atendimento dos clientes de cada rota, as janelas de tempo de todos são respeitadas, incluindo a do depósito, viabilizando

a solução. O número de veículos utilizados é a quantidade de rotas formadas e a distância total percorrida da solução é a soma das distâncias percorridas em particular por cada rota, no total de 45 unidades.

Assim como a maioria dos problemas de roteamento variantes do PRV, o PRVJT também é classificado como um problema NP-Difícil (LENSTRA; RINNOOY KAN, 2006). Consequentemente, apenas soluções em instâncias de ordem reduzida podem ser encontradas por algoritmos exatos. Sugerindo assim que o problema seja tratado por estratégias heurísticas, buscando uma solução aproximada viável em tempo polinomial.

2.3.1 Modelo Matemático para o PRVJT

O Problema de Roteamento de Veículos com Janela de Tempo (PRVJT), conforme Tan et al. (2001) é uma variante do PRV, na qual a cada cliente é associado uma janela de tempo $[e_i, l_i]$ que determina um intervalo de tempo o qual um cliente i pode ser atendido por um veículo k. Cada janela de tempo dispõe de dois limitantes, um inferior e_i e outro superior l_i, que corresponde ao instante de abertura da janela de tempo e instante limite em quem um cliente pode ser atendido. Caso um veículo k chegue antes do tempo determinado pelo instante de tempo e_i para atender um cliente, deve se esperar um tempo w_i para atender o cliente até a abertura da janela de tempo. Também considera-se um tempo s_i para que o atendimento a um cliente i seja atendido. A quantidade de veículos disponíveis é igual a $k = |k|$, em que $K = \{1, \ldots, k\}$ é o conjunto dos veículos. A solução para esse problema é obtida buscando minimizar a distância total percorrida pelos veículos respeitando as restrições impostas pela janela de tempo de cada cliente. Para Cordeau et al. (2001) uma solução possível para o problema não viola nenhuma das restrições do problema, essa é dita solução factível e caracteriza um problema de janelas firme.

O modelo matemático formulado Tan et al. (2001) assim como no PRV original como um grafo completo e não-direcionado $G = (V, A)$, dispondo de um conjunto V de vértices $V = \{0, 1, 2, \ldots, N\}$, onde é o número de pontos a serem visitados, dentre eles cliente e depósito, é dado por N. O índice do depósito é 0. O conjunto A de arestas definido por $A = \{(i, j) \mid i, j \in V, i \neq j\}$ representa as ligações entre os vértices, em que cada uma das arestas (i, j) tem um custo de viagem c_{ij} associado, equivalente à distância entre dois clientes i e j definida por $d_i j$. A variável de decisão x_{ijk} estabelece que um cliente i precede um cliente j atendido por um cliente k, essa variável tem valor igual a 1 caso o cliente i preceda um cliente j e valor 0 se o contrário acontece. A capacidade de cada veículo é igual a Q e r_k é o tempo máximo de atendimento que a rota de um veículo k pode levar para atender os clientes designados a ela. O modelo é denotado por:

$$min \sum_{i=0}^{N} \sum_{\substack{j=0 \\ j \neq i}}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{2.9}$$

Sujeito a:

$$\sum_{j=1}^{N} \sum_{k=1}^{K} x_{ijk} \leq K, i = 0 \tag{2.10}$$

$$\sum_{j=1}^{N} x_{ijk} = \sum_{j=1}^{N} x_{ijk} \leq 1, i = 0, \forall k \in \{1, \ldots, K\} \tag{2.11}$$

$$\sum_{k=1}^{K} \sum_{\substack{j=0 \\ j \neq i}}^{N} x_{ijk} = 1, i \in \{1, \ldots, N\} \tag{2.12}$$

$$\sum_{k=1}^{K} \sum_{\substack{j=0 \\ j \neq i}}^{N} x_{ijk} = 1, j \in \{1, \ldots, N\} \tag{2.13}$$

$$\sum_{i=1}^{N} q_i \sum_{\substack{j=0 \\ j \neq i}}^{N} x_{ijk} \leq Q, k \in \{1, \ldots, K\} \tag{2.14}$$

$$\sum_{i=0}^{N} \sum_{\substack{j=0 \\ j \neq i}}^{N} x_{ijk}(t_{ij} + s_i + w_i) \leq r_k, k \in \{1, \ldots, K\} \tag{2.15}$$

$$t_0 = s_0 = w_0 = 0 \tag{2.16}$$

$$\sum_{k=1}^{K} \sum_{\substack{j=0 \\ j \neq i}}^{N} x_{ijk}(t_{ij} + s_i + w_i) \leq t_j, j \in \{1, \ldots, N\} \tag{2.17}$$

$$e_i \leq (t_i + w_i) \leq l_j, i \in \{1, \ldots, N\} \tag{2.18}$$

$$x_{ijk} \in \{0, 1\} \tag{2.19}$$

O modelo tem como função objetivo (2.9) minimizar o custo das distâncias percorridas pela rota de cada veículo utilizado. A restrição (2.10) impõe que todos os veículos têm como ponto de partida o depósito e que a quantidade máxima de veículos utilizados é igual a K. A restrição (2.11) assegura o retorno dos veículos ao depósito. As restrições (2.12) e (2.13) certifica a continuidade da rota, levando cada veículo a sair de um cliente atendido após o término do serviço. A restrição (2.14) impede que a carga total de um veículo exceda sua capacidade. A restrição (2.15) determina que o tempo

total gasto por um veículo para atender os clientes de uma rota não seja maior do que o limite r_k. A restrição (2.16) precisa que os tempos iniciais de viagem, serviço e espera são iguais a 0. A restrição (2.17) obriga que o tempo de viagem de um cliente i para um cliente j gasto por um veículo k não possa ser maior do que o limitante superior da janela de tempo l_j do cliente j. A restrição (2.18) é responsável por garantir que as restrições das janelas de tempo dos clientes não sejam violadas. A restrição (2.19) define os valores que a variável x_{ijk} pode ter.

2.4 METAHEURÍSTICA GRASP

A metaheurística *GRASP – Greedy Randomized Adaptive Search Procedure* foi proposta por Feo e Resende (1995) e desde então tem sido empregado em vários problemas de otimização combinatória (FESTA; RESENDE, 2009). O *GRASP* se trata de um processo iterativo multipartida que produz aleatoriamente soluções iniciais diferentes, seguido de um procedimento de busca local com o objetivo de obter ótimos locais (FEO; RESENDE, 1995), (RESENDE; RIBEIRO, 2003), sem fazer o uso memória entre suas iterações.

Figura 3 – *GRASP* (Adaptado, ALMEIDA, 2014).

```
1:  procedimento GRASP(α)
2:      f(s*) ← +∞
3:      enquanto não CritérioParada faça
4:          s1 ← FaseConstrutiva(α)
5:          s2 ← BuscaLocal(s1)
6:          se f(s2) < f(s*) então
7:              s* ← s2
8:          fim se
9:      fim enquanto
10:     retorne s*
11: fim procedimento
```

Portanto, a cada iteração o *GRASP* é constituído de duas fases:

- Construtiva: constrói uma solução factível para o problema de forma gulosa-aleatória.
- Busca local: busca melhorar a qualidade da solução gerada na fase anterior.

O processo de busca, por meio das duas fases, se repete até que um critério de parada seja alcançado, esse na maioria das vezes é um número máximo de iterações.

A Figura 3 ilustra o processo na sua forma padrão, nesse uma solução é obtida na fase de construção, solução essa que em seguida é a entrada da próxima fase, a fase de busca local. O procedimento guarda a melhor solução encontrada dentre todas as iterações.

2.4.1 Fase Construtiva

O *GRASP* é um método heurístico que combina características interessantes de procedimentos gulosos e aleatórios para construir soluções viáveis na sua fase construtiva, funcionado de forma iterativa, gulosa-aleatória e adaptativa.

Figura 4 – Fase construtiva do *GRASP* (Adaptado, ALMEIDA, 2014).

```
1: procedimento FaseConstrutiva(α)
2:     Inicializar a lista de candidatos LC
3:     s ← ∅ {Inicialmente a solução está vazia}
4:     enquanto LC ≠ ∅ faça
5:         LRC ← ConstruirLRC(LC, α)
6:         e ← SelecionarAleatorio(LRC)
7:         s ← s ∪ {e}
8:         Atualizar a lista de candidatos LC
9:     fim enquanto
10:    retorne s
11: fim procedimento
```

Nessa fase uma solução é construída iterativamente acrescentando um elemento por vez à solução em construção, dita solução parcial do problema, até que esta esteja completa e factível ao problema. Cada elemento a ser adicionado à solução parcial é escolhido de forma aleatória através de uma Lista Restrita de Candidatos (LRC), a qual contém os elementos a compor o problema que possuem o melhor custo incremental, dentre a Lista de Candidatos (LC) que contém todos os elementos possíveis a integrar uma solução completa do problema. Isto representa o aspecto guloso-aleatório do método. A característica adaptativa está na escolha do próximo elemento escolhido a compor uma solução parcial ser influenciado pelas escolhas anteriores, já que uma vez selecionado a fazer parte da solução parcial, esse é removido da LC, a qual é atualizada e os custos incrementais recalculados (ALMEIDA, 2014). A Figura 4 apresenta um pseudocódigo da fase construtiva do *GRASP*.

Segundo Lima Júnior (2009) há duas estratégias para se construir uma LRC: baseada na qualidade dos elementos que a compõem e baseado na cardinalidade. Para compreender melhor essas duas estratégias considere: uma função de minimização $f(s)$; uma função gulosa $g(s)$ que denota o custo $G_{min} = min\{g(c)|c \in \text{LC}\}$ e $G_{max} = max\{g(c)|c \in \text{LC}\}$ são respectivamente, o menor e o maior custo incremental dos elementos e um parâmetro $\alpha \in [0, 1]$.

- Estratégia de construção com base na cardinalidade:

  ◦ Constrói-se uma lista com de candidatos (LC) formada por elementos que não compõem a solução parcial.

  ◦ Calcula-se as contribuições de cada elemento $c \in$ LC para a solução por meio da função $g(c)$.

  ◦ Ordena-se de forma decrescente os elementos quanto a sua contribuição calculada de modo que o primeiro elemento seja $g(c) = G_{min}$, e o último $g(c) = G_{max}$, ou seja, o que contém a menor contribuição no processo de minimização e o com a maior respectivamente.

  ◦ Acrescenta-se os p primeiros elementos da LC na LRC, o valor de p é dado por:

$$p = 1 + \alpha(N - 1) \tag{2.20}$$

  Onde N é o número total de candidatos.

- Estratégia de construção com base na qualidade dos elementos

Nessa estratégia a construção da LRC considera os valores atribuídos a cada elemento $c \in$ LC, onde o conjunto de elementos que farão parte da LRC são caracterizados por:

$$LRC = \{c \in LC|g(c) \leq G_{min} + \alpha(G_{max} - G_{min})\} \tag{2.21}$$

Assim podemos ver que o conjunto de elementos da LC que constituirão a LRC são aqueles que o seu valor na função $g(c)$ respeitam a condição descrita anteriormente.

Nas duas estratégias de construção da LRC, se o valor de α é igual a 0, o algoritmo é exclusivamente guloso, quando apenas o elemento da LRC com menor custo incremental pode ser selecionado. Caso o parâmetro α tenha valor igual a 1, o algoritmo é totalmente aleatório, já que a LRC formada conterá todos os elementos possíveis, resultando em uma escolha completamente aleatória.

A aleatoriedade do algoritmo na geração da solução inicial assegura que mínimos locais diferentes possam ser obtidos. Dessa maneira, a fase construtiva é responsável pela exploração/diversificação da busca no espaço de soluções (RESENDE; RIBEIRO,

2003). O aspecto probabilístico está nas escolha aleatória de um dos elementos da LRC não ser necessariamente o melhor. O controle entre aleatoriedade e gula é feito pelo valor do parâmetro α (LIMA JÚNIOR, 2009).

2.4.2 Fase de Busca Local

Após a fase construtiva do método *GRASP* não se pode garantir que um ótimo local seja encontrado. Com isso, a fase de busca local é um procedimento que visa aprimorar uma solução obtida inicialmente por um método construtivo, realizando modificações na solução obtida até que se encontre uma solução melhor. Essa fase equivale a explotação/intensificação do processo.

Desse modo, a partir de uma solução *S*, através de modificações realizadas nessa, busca-se obter uma solução *S'* melhor. As modificações são denominadas de movimentos e são realizados em uma busca em vizinhança.

Um algoritmo de busca local divide-se em três passos para realizar a busca em uma vizinhança: a partida, uma solução produzida por um método construtivo; a iteração, onde a solução corrente é substituída sucessivamente por uma melhor solução encontrada; e a parada, que acontece quando o último ótimo local é encontrado.

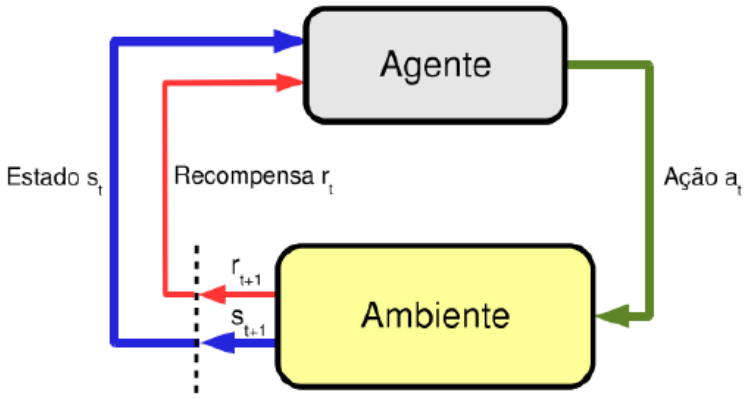
A busca pode ser implementada por meio de duas estratégias: *first-improving*, que substitui a solução corrente quando a primeira melhor solução é encontrada, ou ainda a estratégia *best-improving*, a qual analisa toda uma vizinhança para que a solução corrente possa assumir o melhor vizinho encontrado.

Segundo Feo e Resende (1995) o desempenho de um algoritmo de busca local é dependente de vários aspectos, dentre os quais podemos citar: estrutura da vizinhança da solução, eficiência da avaliação da função de custo e a qualidade da solução inicial. Os autores ainda destacam que a fase construtiva da metaheurística *GRASP* é relevante relacionado ao último aspecto citado, uma vez que um algoritmo de construção que obtenha soluções de boa qualidade pode reduzir o tempo gasto pelo algoritmo de busca local.

2.5 APRENDIZADO POR REFORÇO

A aprendizagem por reforço – AR (em inglês, *Reinforcement Learning* – RL) consiste em um formalismo da Inteligência Artificial (SUTTON; BARTOR, 1998). De acordo com Lima Júnior (2009) um paradigma computacional de aprendizagem em que um agente aprendiz busca, por tentativa e erro, ou seja, por experimentação direta, atingir seu objetivo maximizando uma medida de desempenho baseada em recompensas que recebe ao interagir com um ambiente desconhecido. A Figura 5 mostra como o agente interage com o ambiente e o processo de aprendizado.

Figura 5 – Interação agente-ambiente na aprendizagem por reforço (LIMA JÚNIOR, 2009).



Na Figura 5 a cada instante de tempo t, o agente recebe informações do estado atual do ambiente e baseado nessas, executa uma ação. Esta produz dois resultados: uma recompensa imediata diante da escolha tomada e o processo avança para um novo estado, de acordo com uma determinada distribuição de probabilidades. No passo seguinte $t + 1$ o agente recebe informações do ambiente então alterado e irá decidir dentre um conjunto de ações qual irá escolher.

Uma das características mais importante da AR é não precisar de modelos do ambiente, isto é, o agente se adapta de forma autônoma a ambientes desconhecidos e dinâmicos. Tendo assim seu uso indicado quando modelos a priori não estão disponíveis, ou ainda não há padrões capazes de representar fielmente cenários em que o agente aprendiz irá enfrentar. Na AR o agente interage diretamente com o ambiente no objetivo de executar uma política ótima de ações que o faça alcançar suas metas. Desta forma o método faz um mapeamento de estados em ações de modo a elevar o valor numérico relacionado as recompensas recebidas, sem que o agente aprendiz precise conhecer as ações a serem tomadas, mas encontrar aquelas que o fazem conseguir maiores valores. (ALMEIDA, 2014).

Além do agente e o ambiente pode se considerar quatro elementos básicos da AR (ALMEIDA, 2014) a saber:

- Política: define o padrão de comportamento do agente aprendiz, ou seja, como esse deve decidir, em um dado instante de tempo, entre quais ações tomar e quais serão desprezadas. Seja S o conjunto de estados do ambiente e $A(s)$ o conjunto de ações disponíveis para $s \in S$. Uma política π é uma função de mapeamento de estados em ações:

$$a = \pi(s), \forall a \in A(s), s \in S, \tag{2.22}$$

  isto é, dado um estado s a política indica qual a ação a o agente deve escolher.

- Função de recompensa: determina o objetivo em um problema de AR. Ela relaciona cada estado (ou par estado-ação) a um sinal numérico (recompensa) designando o quanto o estado é atrativo. Sendo assim, o agente tem como objetivo a maximização da quantidade total de reforços recebidos ao longo da execução, denominado retorno acumulado. Para que o valor do retorno não seja muito grande um fator de desconto $\gamma(0 \leq \gamma \leq 1)$ é inserido para diminuir gradualmente valores futuros.

- Função valor: é obtida fazendo o uso dos reforços atual e futuro. Enquanto a função de recompensa atribui um sinal repentino às boas decisões, a função valor indica uma estimativa do ganho total que pode ser acumulado pelo agente durante o aprendizado, tendo início em um estado s e considerando os estados que o sucedem. Por exemplo, um estado pode sempre produzir pequenas recompensas imediatas, mas ainda assim ter um alto valor, pois esse é regularmente frequentado por outros estados que produzem altas recompensas. Quando essa função leva em consideração apenas o estado s, ela é conhecida por função valor-estado, e denotada por $V(s)$:

$$V(s) = E\{r_{t+1} + r_{t+2} + \ldots + r_{t+n} \mid s_t = s\}, \tag{2.23}$$

  onde E é o valor total estimado dos retornos acumulados pelo agente adotando uma política π, a partir de um estado $s_t = s$, no instante t. Em casos em que a função valor considera as ações disponíveis para um certo estado, esta é chamada de função valor estado-ação e é denotada por:

$$Q(s,a) = E\{r_{t+1} + r_{t+2} + \ldots + r_{t+n} \mid s_t = s, a_t = a\}, \tag{2.24}$$

  onde o ganho esperado é dependente da ação escolhida anteriormente.

- Modelo do ambiente: o modelo representa o comportamento do ambiente, isto é, dado: estado e ação, o modelo pode anteceder o estado e a recompensa seguinte de acordo com a probabilidade de transição.

2.5.1 Processos de Decisão Markovianos

O domínio do problema ou tarefa de AR deve ser modelado como um processo de decisão de Markov, tipo de processo esse para o qual os algoritmos de AR foram desenvolvidos e são aplicados. Um Processo de Decisão de Markov – PDM (Markov Decision Process – MDP) é um processo de decisão sequencial e , segundo Puterman (2005), é definido como uma quíntupla $M = \{T, S, A, R, P\}$, cujos seus componentes são:

- T: conjunto discreto de instantes de tempo onde as decisões são tomadas, chamados de instantes de decisão.

- S: conjunto de estados do ambiente que o processo pode assumir.
- A: conjunto de ações possíveis de serem escolhidas em cada instante de tempo. Em $A(s)$ temos as ações possíveis quando o processo se encontra em um estado s, de modo que $A = \bigcup_{A(s)}$, $\forall s \in S$ é a união de todas as ações disponíveis de serem tomadas pelo agente em qualquer estado s.
- $R : S \times A \rightarrow \mathbb{R}$: função de retorno que designa como consequência da escolha de uma ação $a \in A(s)$ uma recompensa $r_t\ (s, a)$, estando em um estado $s \in S$ no instante de decisão $t \in T$
- $P : S \times A \times S \rightarrow [0, 1]$: função de probabilidades de transição, de maneira que a função $p_t\ (s' \mid s, a)$ equivale à probabilidade de que o processo passe para um estado $s' \in S$ no próximo instante, estando no estado s no instante t, escolhendo a ação $a \in A(s)$ a ser executada.

Em um PDM o conjunto de ações possíveis, os retornos e as probabilidades de transição dependem do estado e ação corrente e não dos estados já percorridos e ações tomadas anteriormente. Com isso a probabilidade de um estado s ir a um estado s', dado um par estados-ação (s, a), respeita a propriedade markoviana e pode ser expressa como:

$$P^a_{s,s'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}, \tag{2.25}$$

onde Pr representa a probabilidade do estado s_{t+1} se tornar s', toda vez que a ação a_t for a e o estado s_t for s no instante de decisão t. Tal propriedade tem sua relevância para a AR, já que essa torna viável que decisões sejam tomadas em função apenas do estado corrente, o que significa dizer que um estado futuro do processo é dependente somente do estado em que o processo se encontra e da ação escolhida no momento.

2.5.2 Algoritmo Q-Learning

O algoritmo *Q-Learning* foi proposto por Watkins (1989) e é uma das mais relevantes contribuições na aprendizagem por reforço além de ser uma das técnicas mais populares. O *Q-Learning* é classificado como um método de diferença temporal *off-policy*, visto que sua convergência para valores ótimos de Q não é dependente de nenhuma política usada, isto é, a função ação-valor Q se aproxima em direção da função ação valor ótima $Q*$, por meio de atualizações dos pares estado-ação realizadas conforme os pares são visitados. A expressão de atualização do valor de Q no algoritmo *Q-Learning*, é denotada por:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_t(s_t, a_t) + \gamma max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big] \tag{2.26}$$

onde $s_t = s$ é o estado atual, $a_t = a$ é a ação realizada no estado s_t, r_t (s_t, a_t) é o retorno recebido após a_t ser executada em s_t, s_{t+1} é o estado seguinte, γ é o fator de desconto $(0 \leq \gamma < 1)$ e α, $(0 < \alpha < 1)$, é o coeficiente de aprendizagem. A função $Q(s, a)$ é o valor atribuído ao par estado-ação (s, a) e representa o quão boa é a escolha na maximização da função valor.

Um relevante aspecto desse método é a estratégia de diversificação/intensificação, na qual se deve decidir entre aprender ou usar o conhecimento já adquirido. No algoritmo *Q-Learning* é comum utilizar uma política de decisão ε-gulosa, a qual o método tem comportamento guloso-aleatório, de modo que escolha entre intensificar sua busca, tendo como base a melhor informação até o momento, ou ainda diversificar, explorando novos estados de maneira que informações inéditas possam ser obtidas e posteriormente utilizadas. Tal política tem a "gula" e aleatoriedade controladas pelo valor do parâmetro ε. A atuação da política de modo aleatório têm probabilidade ε e quando gulosa $(1 - \varepsilon)$.

Em sua implementação o algoritmo a cada iteração, chamada de episódio, o agente aprende a melhor decisão a ser tomada na escolha de uma ação. O algoritmo apresentado na Figura 6 é baseado em Watkins (1989).

Figura 6 – Algoritmo Q-learning (Adaptado, LIMA JÚNIOR, 2009).

1: **procedure** QLEARNING($r, \alpha_q, \varepsilon, \gamma$)
2: Inicialize $Q(s, a)$
3: **repeat**
4: Inicialize s
5: **repeat**
6: Selecione a de acordo com a política ε-gulosa
7: Observe os valores de r e s'
8: $Q(s,a) \leftarrow (1 - \alpha_q)Q(s,a) + \alpha_q(r + \gamma \max_{a \in A}(Q(s', a))$
9: $s \leftarrow s'$
10: **until** Encontrar um estado final
11: **until** Atingir NEp episódios
12: **return** $Q(s, a)$
13: **end procedure**

2.6 HIPERHEURÍSTICAS

As hiperheurísticas se referem a qualquer método heurístico que gerencie (administre) outras heurísticas na resolução de um determinado problema de otimização. Essas propõem resolver problemas computacionais difíceis com um novo nível de abstração, em que um conjunto de heurísticas específicas são combinadas e parametrizadas buscando equilibrar as vantagens e desvantagens de cada procedimento, minimizando os erros de decisão de cada um, onde se escolhe o cenário mais apropriado para aplicação

de uma dada heurística dentro de um processo de otimização em que há a intercalação de heurísticas. Essa tarefa é feita idealmente sem nenhuma informação direta do domínio do problema abordado, o que possibilita que tal estratégia seja empregada em vários problemas diferentes sem que grandes modificações sejam feitas, provendo facilidade de aplicação e reuso (SUCUPIRA, 2007).

Operando em alto nível, lidando indiretamente com o problema, essa estratégia flexível pode obter soluções de boa qualidade sem que muito esforço seja realizado para analisar e adaptar o algoritmo a um novo problema a ser tratado. Essa característica de uma abordagem "caixa-preta" também é presente nas metaheurísticas, porém se distingue pelo seguinte fato: as abordagens metaheurísticas realizam suas buscas no espaço de soluções-candidatas do problema, enquanto as hiperheurísticas operam em um espaço de busca do conjunto de heurísticas específicas, que lidam explicitamente com informações do problema, denominadas heurísticas de baixo nível (MORENO, 2012).

Uma arquitetura bem conhecida e já aplicadas em trabalhos anteriores é a de Ross et al. (2003), que leva em consideração todas as características já citadas antes. Segundo Sucupira (2007) os métodos hiperheurísticos já desenvolvidos na sua maioria tem como características:

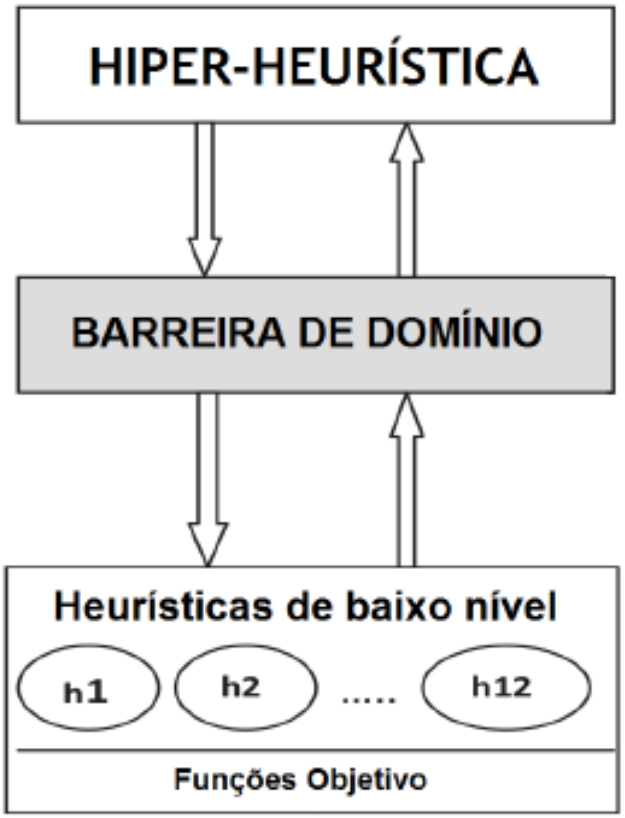
- Uma solução para a instância do problema tratado
- Um conjunto de heurísticas simples capazes de produzir soluções
- Uma função capaz de avaliar as soluções

Dessa maneira pode-se notar que tais estratégias são algoritmos que podem ser aplicáveis a qualquer domínio de problema, requerendo somente a definição do nível mais baixo, dotando ele de um conjunto de heurísticas específicas que possam lidar com o problema.

Com base nessa ideia, outro conceito importante a ser tratado é o de barreira de domínio que tem como objetivo garantir que uma hiperheurística possa ser aplicada a qualquer problema de otimização, onde o método tem conhecimento apenas de informações não atreladas ao domínio do problema. Assim algumas informações a serem conhecidas pelo método podem ser: quantidade de heurísticas de baixo nível, medição do desempenho dessas heurísticas, estatísticas, melhoria obtidas, parametrizações, etc. A Figura 7 apresenta essa estrutura.

Como podemos observar na Figura 7, há uma estrutura em três camadas: a hiperheurística que opera em alto nível, a barreira de domínio a qual garante um fluxo de informações não ligadas diretamente ao problema e o nível mais baixo, contendo as heurísticas de baixo nível, função de avaliação ou qualquer estrutura e operador que necessite de conhecimento específico do problema tratado.

Figura 7 – Barreira de domínio (Adaptado, BURKE et al., 2010a).



2.6.1 Histórico

Apesar do termo hiperheurística ter sido introduzido por Cowling et al. (2001) apenas no século XX, em outros trabalhos, como Ross et al. (2003), já eram observadas que estratégias gerenciadoras de heurísticas e a utilização dinâmica de heurísticas já eram utilizadas na década de 1980 em procedimentos de planejamento automático como o sistema *PRODYGY* (MINTON, 1988 apud ROSS et al., 2003), que por meio de aprendizagem automática evoluía processos de decisão.

O uso de estratégias explícitas de seleção de heurísticas é da década de 1990, entre essas podemos citar o escalonador LR-26, parte do sistema *COMPOSER* (GRATCH et al., 1993), que tratava um problema de programação inteira binária, inserido no planejamento de comunicações entre satélites artificiais e três estações em terra. Seu algoritmo tinha 5 fases e 51 estratégias disponíveis, onde o processo de seleção teve eficiente desempenho.

Fang et al. (1994) apresentaram um método que usava um algoritmo genético para escolher heurísticas a serem aplicadas a um problema de escalonamento (*open-shop scheduling*) denominado *Evolving Heuristic Choice*. Algumas das heurísticas dispostas

eram: selecionar a tarefa com menor tempo de processamento e selecionar a tarefa com maior tempo de processamento. O método atingiu bons resultados em várias instâncias conhecidas e em algumas, resultados superiores aos existentes. Apesar disso duas desvantagens encontradas foram o consumo de recursos e sua aplicação ser restrita ao domínio do problema.

Terashima-Marín et al. (1999) elaboraram um Algoritmo Genético(AG) para um problema de construção de horários para exames acadêmicos, os quais poderiam ocorrer em alguns intervalos de tempo fixos em salas de diversos tamanhos. Nesse problema várias restrições eram consideradas como: dois exames não podem ocorrer ao mesmo tempo se algum estudante realizará ambos, dois exames podem acontecer na mesma sala, ao mesmo tempo, desde que a sala tenha capacidade, alguns exames têm suas próprias restrições de horários, é desejável que um aluno nunca faça dois exames seguidos e que os exames de maior porte ocorram antes.

Para resolver esse problema o método proposto por Terashima-Marín et al. (1999) tinha o intuito de estabelecer quais estratégias iriam compor um segundo algoritmo que iria lidar com o problema diretamente em duas etapas. Na primeira há a aplicação de duas heurísticas, uma heurística H_1 que seleciona um evento, em seguida uma heurística H_2 que define o horário que o evento acontecerá, isso se repete até que uma condição X seja alcançada. Na segunda etapa a condição de parada muda e as heurísticas H_1 e H_2 são trocadas respectivamente pelas heurísticas H_3 e H_4. O algoritmo genético tinha o objetivo de escolher as quatro heurísticas, além da condição de parada X. O método obteve resultados satisfatórios, mesmo em instâncias de grande porte.

2.6.2 Motivação

Algoritmos personalizados baseados em estratégias metaheurísticas mesmo com sua eficiência comprovada em vários problemas de otimização tem sua implementação na prática limitada a um único domínio do problema. Adaptar um algoritmo apropriado a um domínio de problema para outro requer modificação das estruturas de vizinhança e novas parametrizações. Para que essas alterações sejam feitas é necessário conhecimento específico do problema e da estratégia adotada. Ou mesmo em um único domínio de problema uma heurística/metaheurística pode ter um bom desempenho em algumas instâncias, já em outras, resultados não tão satisfatórios (MORENO, 2012).

Bai e Kendall (2005) e Chakhlevitch e Cowling (2008) perceberam que essas restrições requerem atenção quando os dados do problema e requisitos do negócio se alteram constantemente, exigindo um maior esforço em manter a estratégia adequada. Ross et al. (2003) faz críticas quanto a aplicação prática das metaheurísticas, apontando que os algoritmos mais usados atingiram um elevado nível de complexidade, sugerindo que é mais comum a utilização de métodos de busca simples, mesmo que resultados

inferiores sejam encontrados.

As hiperheurísticas são técnicas emergentes e estratégias interessantes quando a intenção são métodos genéricos, que se adaptem a instâncias bem distintas de um domínio de problema ou ainda quando um método precisa ser aplicado a um domínio de problema semelhante, porém diferente. Podemos assim ressaltar sua flexibilidade, facilidade de aplicação e reuso a um problema de otimização em específico ou a vários com algum aspecto em comum.

2.6.3 Classificação

Uma classificação dos métodos hiperheurísticos existentes pode ser encontrada em Chakhlevitch e Cowling (2008), essa se define nas seguintes categorias:

- Baseadas em seleção aleatória: selecionam aleatoriamente as heurísticas de baixo nível dentre um conjunto de métodos disponíveis.
- Gulosas: executa todas as heurísticas de baixo nível e a cada iteração escolhe aquela que obteve a melhor solução.
- Baseadas em metaheurísticas: utilizam de uma estratégia metaheurística para escolher quais heurísticas de baixo nível serão aplicadas.
- Com aprendizado: utilizam alguma técnica de aprendizado para selecionar heurísticas de baixo nível, baseado no desempenho histórico da aplicação dessas.

Essa categorização pode ser ambígua em alguns casos, e ser carente para precisar a qual categoria um método hiperheurístico pode fazer parte. Uma outra classificação, essa mais genérica, pode ser vista em Ozcan, Uyar e Burke (2009), que dividiram as hiperheurísticas em dois grupos:

- Hiperheurísticas que escolhem heurísticas
- Hiperheuríticas que geram ou combinam heurísticas

Esse ponto de vista foi evoluído por Burke et al. (2010a) que propuseram que tais métodos fossem classificados pela análise de duas dimensões:

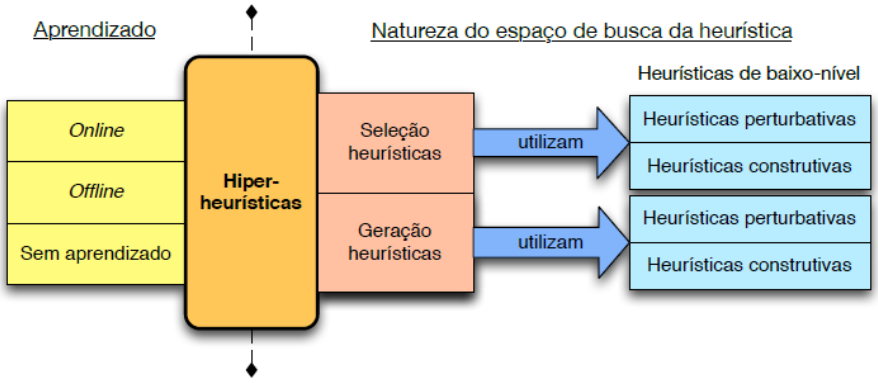
- Pela natureza do espaço de busca
- Pela estratégia de aprendizado

Na primeira dimensão podemos admitir dois níveis. O primeiro nível considera duas categorias de hiperheurísticas: os métodos baseados em seleção de heurísticas e aqueles que geram heurísticas. Já o segundo nível diferencia as estratégias em questão

quanto as classes das heurísticas de baixo nível selecionadas ou geradas, sendo essas construtivas: que constroem uma solução factível para um problema passo a passo analisando em cada um que elemento deve compor a solução; ou perturbativas: através de movimentos que alteram uma solução válida e podem gerar novas soluções possíveis.

Na segunda dimensão de análise a classificação separa as hiperheurísticas quanto a estratégia de aprendizado, dividindo-se em: aprendizado online, o aprendizado ocorre durante o processo de busca de forma instantânea, baseada em métricas e sem um treinamento prévio, aprendizado offline que necessita de treinamento prévio e toma decisões baseadas nesses sem nenhuma atualização do conhecimento; e sem aprendizado onde não há nenhuma ferramenta de aprendizado. Na Figura 8 temos o esquema dessa classificação.

Figura 8 – Classificação das hiperheurísticas (Adaptado, BURKE et al., 2010a).



3 TRABALHOS RELACIONADOS

3.1 ABORDAGENS PARA O PRVJT

3.1.1 Métodos exatos

Os métodos exatos são estratégias que garantem a obtenção da melhor solução (ótima) para um problema, dentre todas as possíveis. Tais métodos exigem um grande esforço computacional, pois exploram todo o espaço de busca de forma exaustiva. Esses geralmente são eficientes apenas para instâncias de pequeno e médio porte, não sendo capaz de oferecer soluções para problemas tratados em instâncias maiores, onde há um alto grau de combinações (como problemas da classe NP-difícil) que geram um imenso espaço de busca a ser percorrido, demandando tempo e recursos computacionais inviáveis.

Na literatura para o PRVJT existe alguns métodos que utilizam essa abordagem para resolvê-lo, tais métodos são dotados de técnicas de relaxação em problemas de programação inteira, possibilitando também a combinação de modernas técnicas de plano de corte. Dentre os alguns métodos encontrados temos: *Branch-and-Bound*, *Branch-and-Cut*, *Branch-and-Price* e *Branch-and-Cut-and-Price*.

O algoritmo *Branch–and-Bound* foi o método adotado por Barker (1982) como resolução para um problema de roteamento com restrições de tempo e somente um veículo, e por Kolen et al. (1987) para resolver o PRVJT. Nesse algoritmo de otimização global faz-se a enumeração inteligente de prováveis pontos a solução ótima do problema, considerando o PRVJT um problema de programação inteira mista $0-1$. Nesse método primeiramente todas as restrições de integralidade do problema são relaxadas, em seguida o método se divide em duas partes. Na fase de *branching* (ramificação), uma variável binária é fixada, ora em 0, ora em 1, formando dois subproblemas a serem solucionados isoladamente, sendo o custo da função objetivo da solução ótima de cada um desses subproblemas um limitante superior para o problema original. Aos subproblemas é associado uma estrutura do tipo árvore, a qual tem como raiz o problema em que todas as restrições são relaxadas e cada mudança de nível desta está relacionada a ramificação de uma variável. Os subproblemas não solucionados formam nós pendentes se tornando ramos não visitados da árvore. A fase de *bounding* têm seu início sempre que uma solução ótima satisfaz todas as restrições de integralidade do problema original, nessa fase baseia-se na eliminação dos nós pendentes da árvore de subproblemas, os quais têm valor da função objetivo pior do que a solução ótima encontrada no seu início.

Bard, Kontoravdis e Yu (2002) empregaram o algoritmo *Branch-and-Cut* para resolver o PRVJT. Nesse método há uma combinação do procedimento *Branch-and-Bound*

e técnicas de plano de cortes. Em que passada a resolução do problema relaxado o método de planos de corte é aplicado no intuito de diminuir ainda mais a região factível dos subproblemas associados ao nó da árvore. Tal redução é obtida através do acréscimo de restrições que formam o politopo factível do problema.

O método *Branch-and-Price* foi aplicado ao PRVJT por Desrochers et al. (1992) conseguindo obter solução ótima principalmente para as classes de instâncias testadas que dispunham de um número reduzido de clientes (25 e 50 clientes). O algoritmo *Branch-and-Price* é especialmente um método *Branch-and-Bound* com geração de colunas, onde em cada nó da árvore de busca obtida com o *Branch-and-Bound*, emprega-se o método de geração de colunas com a ideia de atingir diferentes soluções viáveis. A abordagem por geração de colunas consiste em uma generalização da decomposição de Dantzig-Wolfe (1960), a qual separa o problema original em duas parcelas: o problema principal e o subproblema. Devido ao número elevado de soluções fatíveis somente uma pequena porção desse conjunto é considerada no problema principal. Em cada iteração principal do método examina-se a existência de uma rota que possa diminuir a distância total percorrida. Isso é obtido através da resolução de um subproblema, no qual são relaxadas as restrições de capacidade e janela de tempo. Se a solução é viável, sua coluna é adicionada ao modelo e o procedimento se repete até que uma solução ótima seja obtida.

Semelhante ao último método citado Jepsen et al. (2006) usaram o método *Branch-and-Cut-and-Price* como resolução para o PRVJT. Assim como na geração de colunas do *Branch-and-Price* neste também é feito uma divisão do problema em: problema principal e secundário. Adicionado a isso planos de corte são utilizados no problema principal. Essa abordagem conseguiu atingir solução ótima em 10 das 56 instâncias sugeridas por Solomon (1987).

3.1.2 Métodos heurísticos

Classificado por Lenstra e Rinnooy Kan (2006) como problema NP-difícil, o PRVJT na sua alta complexidade de resolução pela enorme quantidade de combinações possíveis para uma solução torna impraticável o uso de métodos exatos em um tempo computacional hábil para maioria de suas instâncias(médio e grande porte). As instâncias de grande porte são aquelas que possibilitam que um problema real seja melhor representado. Como alternativa viável de resolução sugere-se a aplicação de métodos heurísticos e metaheurísticos. A utilização desses métodos não garante a obtenção da solução ótima, porém oferece soluções de boa qualidade, com esforços e tempo computacional reduzido.

Os algoritmos heurísticos, ao contrário dos métodos exatos, são aqueles que exploram uma parte reduzida do espaço de busca, baseados em algum conceito ou modelo de como o problema pode ser tratado ou avaliado. Esse tipo de técnica

eventualmente pode encontrar uma solução ótima, mas geralmente produzem soluções de boa qualidade, limitadas a alcançar ótimos locais. Comumente são aplicadas para gerar soluções iniciais para outros métodos, ou ainda seguidos e combinados de procedimentos especializados. As heurísticas para o PRVJT podem ser classificadas como: construtivas e de busca local.

Heurísticas construtivas

As heurísticas construtivas são métodos capazes de construir, passo a passo, uma solução viável para o problema partindo de uma solução qualquer, geralmente não válida. Em particular para os problemas do tipo PRV as heurísticas construtivas podem ser do tipo: sequencial, aquela onde as rotas são construídas uma por vez, ou ainda paralelas, em que todas as rotas são construídas ao mesmo tempo. Para o PRVJT em particular podemos citar aquelas que não consideram apenas a distância entre clientes para construir as rotas, mas também suas janelas de tempo, ou ainda alguma equação matemática que descreva uma relação entre essas duas características. Como exemplo dessas podemos citar:

- Vizinho mais próximo orientado ao tempo

Proposto por Solomon (1987) essa heurística é uma adaptação do método de economias denominada heurística do vizinho mais próximo orientado ao tempo. Nesse método cada rota inicia adicionando o cliente não roteado mais próximo ao depósito e, a cada iteração seguinte, seleciona-se para compor a rota corrente, o cliente ainda não atendido mais próximo do último cliente acrescentado à rota construída, desde que este respeite todas as restrições do problema como: atendimento da janela de tempo e capacidade do veículo. Caso não exista a possibilidade de incluir nenhum novo cliente a rota corrente, uma nova rota é iniciada. O método se repete até que todos os clientes sejam atendidos.

O modelo que estabelece a proximidade entre os nós i e j, respectivamente o último cliente inserido na rota e qualquer outro cliente ainda disponível, os quais representam o depósito por onde uma rota deve começar e os clientes a serem atendidos, é definido por um custo c_{ij}. Solomon (1987) considera esse custo presente nos arcos do PRVJT baseado em três parâmetros p_1,p_2,$p_3 \geq 0$, de modo que $p_1 + p_2 + p_3 = 1$. Definido os parâmetros, o custo associado ao arco (i, j) é denotado pela equação:

$$\bar{c_{ij}} = p_1 d_{ij} + p_2 t_{ij} + p_3 v_{ij}, \tag{3.1}$$

onde:

$$t_{ij} = b_j - (b_i + s_i), \tag{3.2}$$

$$v_{ij} = l_j - (b_i + s_i + t_{ij}) \tag{3.3}$$

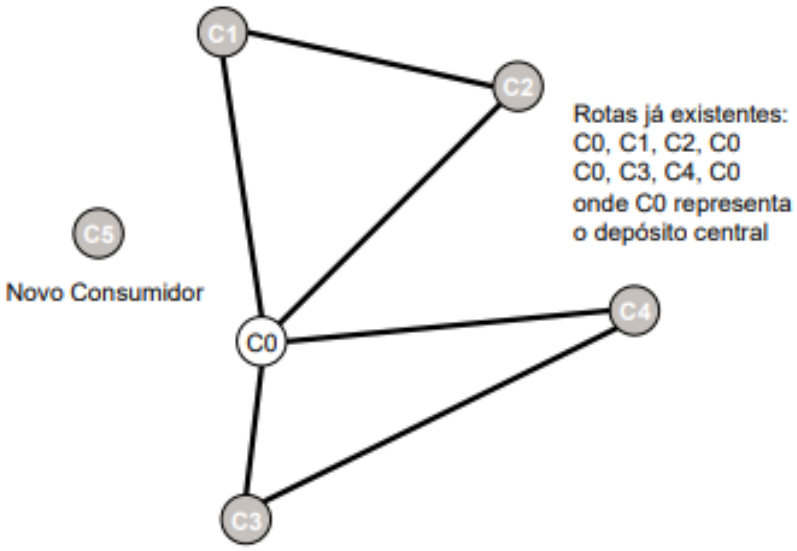
As variáveis que compõem a equação representam d_{ij} : a distância entre os nós, t_{ij} : a diferença entre o serviço finalizado em i e o atendimento em j e v_{ij} : a urgência do atendimento do cliente j, dado pela diferença entre o instante final da janela de tempo em j e o instante de chegada do veículo a este nó.

- *Push-Forward Insertion Heuristic (PFIH)*

Segundo Larsen (1999) o *Push-Forward Insertion Heuristic (PFIH)* é um método construtivo eficiente para calcular o custo da inserção de um novo cliente a uma rota construída. Esse método proposto Solomon (1987) tem seu custo calculado de acordo com: sua posição geográfica, o fim da janela de tempo e o ângulo polar existente entre o cliente e o depósito por onde as rotas são iniciadas.

Considere a Figura 9 como uma solução que antecede a inclusão do cliente C5 em alguma das duas rotas disponíveis.

Figura 9 – Solução atual antes da inserção do cliente C5 (Adaptado, OLIVEIRA, 2007).



A qualidade, assim como a validade de uma rota é verificada através da inserção do cliente C5 em todas as arestas do grafo da solução atual como mostra a Figura 10. Depois a aresta onde a inserção feita apresentar o menor custo a distância total percorrida por todas as rotas e for factível ao PRVJT, respeitando todas suas restrições, é escolhida para ter o cliente inserido como mostra a Figura 11. Quando não há nenhuma aresta possível para se inserir um dado cliente uma nova rota de ser criada.

Figura 10 – De acordo com a Figura 9 as possíveis soluções para inserção do cliente C5 (Adaptado, OLIVEIRA, 2007).

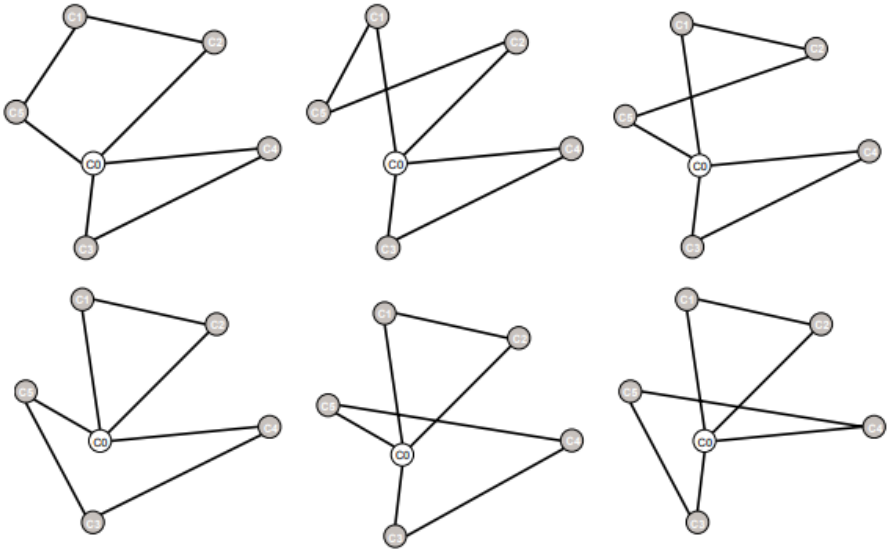
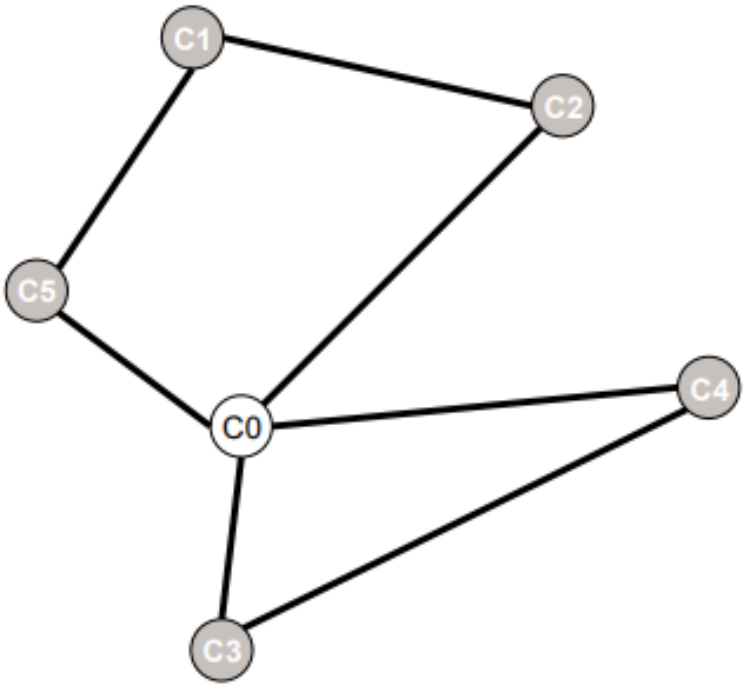


Figura 11 – Solução selecionada para inserção do cliente C5 (Adaptado, OLIVEIRA, 2007).



A ordem com que os clientes são escolhidos para serem inseridos na solução é determinada pela função que calcula o custo de inserção de cada cliente i na solução em construção, está função é denotada por:

$$C_i = -\alpha d_{0i} + \beta l_i + \gamma[(p_i/360)d_{0i}] \tag{3.4}$$

onde:

- $\alpha = 0,7$; $\beta = 0,1$ e $\gamma = 0,2$ (definidos empiricamente em Solomon (1987));
- d_{0i}: distância do depósito central e do cliente i;
- l_i: limite superior da janela de tempo de chegada ao cliente i;
- p_i: ângulo da coordenada polar do cliente i, referente ao depósito central.

Heurísticas de busca local

Os procedimentos de busca local ou de refinamento são técnicas que têm como objetivo aprimorar uma determinada solução através de uma busca que explore sua vizinhança. Entende-se por vizinhança de uma solução o conjunto de soluções alcançáveis através de movimentos de troca de posição ou ordem dos elementos que a constituem. Para esse tipo de método temos duas abordagens. A primeira é denominada *first acept (FA)* ou *first best (FB)*, a qual retorna como solução a primeira melhor solução encontrada em relação a solução atual. A segunda abordagem é chamada de *Best Acept (BA)* ou *Global Best (GB)*, a qual tem como solução a ser retornada aquela que é encontrada depois de que toda a vizinhança de uma solução é verificada. Esse tipo de procedimento geralmente é utilizado para melhorar uma solução inicial oriunda de outro método ou ainda como pós-otimização. Alguns métodos dessa classe são usados em problemas de roteamento, dentre esses pode citar:

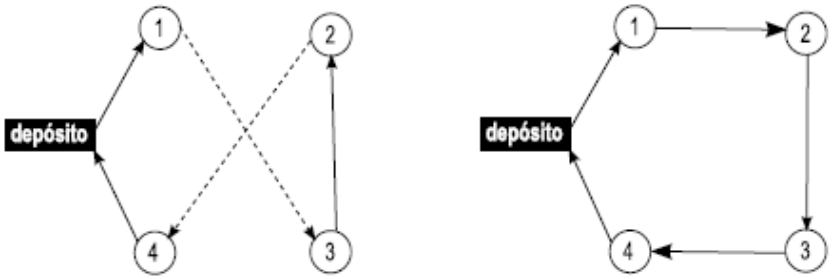
- *K-opt*

Proposto por Lin (1965) esse método foi desenvolvido para o clássico Problema do Caixeiro Viajante (PCV). Essa técnica é um procedimento intra-rota, ou seja, os elementos envolvidos fazem parte de uma mesma rota. Nesta rota é feito a remoção de k arestas e, logo em seguida, seleciona-se k novas arestas, dentre todas as disponíveis, para substituir as arestas anteriormente removidas, formando uma nova rota viável de custo menor que a rota anterior. Na Figura 12 é apresentado um movimento *2-opt*, onde $k = 2$. Podemos perceber que as duas arestas removidas foram substituídas por duas novas arestas, após verificadas todas as possibilidades de reconexão da rota. No PRVJT a substituição das duas novas arestas deve respeitar todas as restrições impostas pelo problema, formando assim sempre uma rota viável.

- *2-opt**

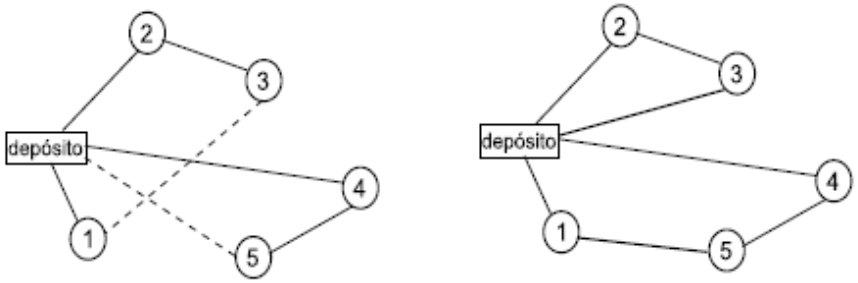
Baseando-se no método criado por Lin (1965), Potvin e Rousseau (1995) propuseram uma adaptação desenvolvendo uma técnica denominada *2-opt**, um procedimento inter-rotas, onde as arestas envolvidas nas trocas fazem parte de rotas diferentes. O

Figura 12 – Algoritmo *K-opt* (Adaptado, VIEIRA, 2008).



método foi modificado para respeitar as janelas de tempo dos clientes baseado na orientação da rota formada. O *2-opt** começa removendo duas arestas, onde cada uma pertence a uma rota. Logo após, duas novas arestas são selecionadas e inseridas, uma em cada rota, realizando a reconexão das rotas e mantendo suas viabilidades. A Figura 13 mostra as alterações realizadas nas rotas 1 e 2.

Figura 13 – Algoritmo *2-opt** (Adaptado, VIEIRA, 2008).



3.1.3 Metaheurísticas

Metaheurísticas são algoritmos aproximativos que usam de processos iterativos para guiar heurísticas subordinadas pela associação inteligente de diferentes conceitos, visando a exploração/explotação do espaço de busca no interesse de encontrar, de forma eficiente, soluções de alta qualidade. São estratégias não determinísticas as quais não garantem uma solução ótima, porém produzem soluções de alta qualidade à um baixo custo computacional, através de procedimentos de alto nível que mesclam aleatoriedade, memória e gula, permitindo a essas escaparem de ótimos locais. Esse tipo de estratégia explora uma parcela do espaço de soluções de maneira mais ampla e eficiente, o que permite que soluções de melhor qualidade sejam encontradas (BLUM; ROLI, 2003).

Nas últimas décadas uma gama de metaheurísticas foram propostas e aplicadas a diversos problemas reais nas várias áreas das ciências (OSMAN; LAPORTE, 1996), dentre as mais conhecidas podemos citar: Algoritmos Genéticos (HOLLAND, 1975), *Simulated Annealing* (KIRKPATRICK et al., 1983), Busca Tabu (GLOVER, 1986), *Greedy*

Randomized Adaptive Search Procedure - GRASP (FEO; RESENDE, 1995), Otimização por Colônia de Formigas (DORIGO, 1999).

Na literatura há diversas abordagens que tratam o PRVJT utilizando as metaheurísticas já citadas. Tais trabalhos propuseram estratégias que buscavam minimizar os custos empregados no PRVJT, sejam esses em distância total percorrida ou em número de veículos utilizados. Todos os trabalhos a seguir discutidos tiveram seus algoritmos testados com as conhecidas instâncias de Solomon (1987).

Algoritmos Genéticos

Ombuki et al. (2006) apresentaram como proposta para o PRVJT um AG multiobjetivo que contava com um operador baseado na rota de menor custo, o qual era utilizado na fase de recombinação. Denominado *Best Cost Route Crossover (BCRC)* o operador escolhia um par de cromossomos pais P_1 e P_2 e depois da seleção do ponto de corte, feita de forma aleatória, cada pai escolhido teria uma parte do material genético selecionada. Os filhos gerados F_1 e F_2, tem uma parte do seu material genético retirado, sendo a parte removida de F_1 um gene de P_2 e a parte removida de F_2 um gene de P_1. Os genes retirados eram reinseridos de forma aleatória nesses filhos. Os autores usaram ainda na mutação do algoritmo o operador *Constrained Route Reversal Mutation (CRRM)*, o qual tinha o objetivo de fugir de ótimos locais através de um simples processo de inversão, onde o trecho a ser invertido é selecionado aleatoriamente.

Vieira (2008) sugeriu como proposta um estudo de técnicas que dá ênfase aos Algoritmos Genéticos. Neste trabalho foram implementados vários tipos de cruzamentos e técnicas de mutação, além do método *Hill-Climbing*. No método gerava-se a população inicial utilizando a heurística *PFIH* de Solomon (1987) seguida da seleção dos melhores vizinhos oriundos da técnica *2-Interchange*, compondo o restante da população com cromossomos aleatórios. O método proposto seguia o esquema tradicional de um AG repetindo os processos de seleção, cruzamento e mutação, acrescentando o método *Hill-Climbing* e um operador *Recovery* respectivamente. O método *Hill-Climbing* é aplicado após a mutação e também como técnica de pós-otimização sobre o melhor indivíduo encontrado. O operador *Recovery* tem a finalidade de evitar que bons pais saiam da população, onde ao fim de cada iteração uma porcentagem dos piores filhos gerados é substituída por melhores indivíduos da população inicial. Segundo o autor essa porcentagem deve ser pequena e menor que a taxa de mutação, pois impede que o algoritmo convirja muito cedo para um ótimo local.

Simulated Anealing

Dentre os trabalhos que apresentaram uma resolução para o PRVJT utilizando a metaheurística *Simulated Anneling(SA)* podemos citar Thangiah et al. (1994), Chiang e Russell (1996), Czech e Czarnas (2002), Li e Lim (2003) e Woch e Lebkowski (2009).

Thangiah et al. (1994) apresentaram um método híbrido que usava as metaheurísticas *Simulated Annealing* e Busca Tabu para explorar a vizinhança do mecanismo *λ-interchange* proposto. As soluções iniciais eram obtidas com a heurística *Push Forward Insertion Heuristic (PFIH)* de Solomon (1987) e um algoritmo Genético baseado em uma heurística setorial.

Chiang e Russell (1996) também aplicaram o *SA* ao problema de roteamento de veículos e problemas de *Scheduling* com restrições de janela de tempo. Neste trabalho foram implementadas duas estruturas de vizinhança, as técnica *λ-interchange* de Osman (1993) e *k-node interchange* de Christofides e Beasley (1984). Foi proposto um aperfeiçoamento no processo de recozimento, o qual foi dotado de uma memória de curto prazo, em que uma lista tabu era examinada para melhorar o desempenho da metaheurística.

Czech e Czarnas (2002) propuseram uma versão paralela do *SA* usando uma função objetivo bi-critério. Onde os processos paralelos foram utilizados para acelerar o processo de recozimento e para atingir uma melhor precisão nas soluções. Nesse trabalho os processos foram utilizados cooperativamente compartilhando entre si as melhores soluções encontradas a cada passo.

Li e Lim (2003) também utilizaram a heurística *PFIH* de Solomon (1987) para obter soluções iniciais, assim como também uma técnica de *k*-reinicialização no intuito de evitar ótimos locais. No procedimento de busca local foram aplicados três operadores: *shift operator* (operador de deslocamento), *exchange* (operador de troca) e operadores de reinserção baseados nos métodos também utilizados em Salvesberg (1992) e Taillard (1997).

Woch e Lebkowski (2009) apresentaram um *SA* com uma função objetivo a ser minimizada baseada em dois critérios, o número de veículos e a distância total percorrida. Os autores ainda usaram uma heurística construtiva de ordenação para as janelas de tempo dos clientes e uma função de transferência baseada na ideia *"first cluster and then route"* para obter uma solução sequencial, a qual considera primeiro a distância entre os clientes agrupados e depois é feito o roteamento dos mesmos.

Busca Tabu

Schulze e Fahle (1999) e Badeau et al. (1997) desenvolveram uma versão da metaheurística Busca Tabu em paralelo.

Lau et al. (2003) apresentaram uma estratégia Busca Tabu, que contava com uma lista de espera para o problema, onde a quantidade de veículos era fixada e as janelas de tempo dos clientes poderiam ser violadas, mediante penalizações. Essa lista de espera era composta de clientes ainda não roteados, os quais eram recolocados em trocas feitas entre a lista e a solução parcial, como se tal fosse uma rota. A janela de tempo era relaxada para que se pudesse incluir uma quantidade maior de clientes em uma rota.

Barbosa (2005) propôs uma Busca Tabu adaptativa para resolver o PRV clássico e o PRVJT, em que os valores dos parâmetros tabu eram ajustados conforme uma análise de padrões identificados durante a busca.

GRASP

O método *GRASP* foi aplicado ao PRVJT em Kontoravdis e Bard (1995), onde o primeiro objetivo do método era minimizar o número de veículos utilizados, seguido por diminuir a distância total percorrida. Na abordagem utilizava-se na fase de construção duas listas para gerar uma solução viável, essas avaliavam o custo de se inserir um cliente disponível baseado não somente na distância em que estavam dispostas, mas também considerando as restrições de tempo de cada um. Na fase de refinamento para encontrar um ótimo local para a solução proveniente da fase anterior, a abordagem usava de um procedimento de eliminar rotas, primeiramente buscando minimizar o número de veículos e logo após a técnica *2-opt exchange* para minimizar a distância total percorrida.

Franco et al. (2009) propuseram uma adaptação da metaheurística *GRASP* tradicional como resolução para o PRVJT, na qual um novo meta-modelo era baseado na iterações do tradicional método. Nesta um método de geração heurística de colunas operava nas iterações do *GRASP*, utilizando de um Modelo de Particionamento de Conjuntos juntamente a busca local. A adaptação permitia que as soluções pudessem ser armazenadas e empregadas em novas buscas, promovendo assim uma ordenação dos métodos heurísticos aplicados e a capacidade de ajuste de seus parâmetros.

Colônia de Formigas

Gambardella, Taillard e Agazzi (1999) utilizam um algoritmo baseado na metaheurística Colônia de Formigas, denominado *MACS-VRPTW (Multiple Ant Colony System for Vehicle Routing Problem with Time Windows)* – Sistema Múltiplo de Colônia de Formigas para o Problema de Roteamento e Veículos com Janelas de Tempo. Nesta estratégia usava-se duas colônias trabalhando em paralelo, a primeira, *ACS-VEI*, tinha o objetivo de minimizar o número de veículos e a segunda, *ACS-TIME*, buscava reduzir a distância total percorrida. Nesse método quando encontrada uma solução que utilizava uma quantidade menor de veículos, essa solução era repassada para a segunda colônia que tentava minimizar da distância percorrida pelos veículos empregados. O processo é repetido até que nenhuma melhoria acontecesse. As colônias usavam de feromônios distintos e compartilhavam a melhor solução obtida para atualização dos mesmos.

Fraga (2006) também apresentou uma resolução para o PRVJT através de um método híbrido que combinava o algoritmo de Colônia de Formigas e a Busca Tabu, aliado ainda a técnica de intensificação Reconexão por Caminhos (*Path Relinking*).

3.2 ABORDAGENS HIPERHEURÍSTICAS

Diversas abordagens hiperheurísticas na literatura foram propostas como solução para vários problemas de otimização combinatória. Algumas dessas abordagens foram desenvolvidas inspiradas em métodos metaheurísticos tradicionais, dentre as quais podemos citar: hiperheurística baseada na metaheurística Busca Local Iterada (Burke et al., 2010b), hiperheurística inspirada no Algoritmo Memético (Burke et al., 2011) e hiperheurística busca tabu com aceitação adaptativa (Burke et al., 2011). As abordagens discutidas a seguir também foram classificadas segundo os critérios de (Burke et al., 2010a).

Figura 14 – Hiperheurística Iterada (Adaptado, MORENO, 2012).

Algoritmo HiperHeurísticaIterada()

1: Cria uma solução inicial s_0
2: $s \leftarrow s_0$
3: **para** cada heurística h de HeurísticasDeBuscaLocal **faça**
4: $s' \leftarrow ExecutaHeurística(h, s_0)$
5: **se** $f(s') < f(s)$ **então**
6: $s \leftarrow s'$
7: **fim se**
8: **fim para**
9: **repete**
10: $h_{pert} \leftarrow SelecionaUniformeEAleatoriamente$(HeurísticasDePerturbação)
11: $s' \leftarrow ExecutaHeurística(h_{pert}, s)$
12: $candidatas \leftarrow \emptyset$
13: **para** cada heurística h de HeurísticasDeBuscaLocal **faça**
14: $s_{cand} \leftarrow ExecutaHeurística(h, s')$
15: $Adiciona(s_{cand}, candidatas)$
16: **fim para**
17: $s' \leftarrow MelhorSolução(candidatas)$
18: **se** $f(s') < f(s)$ **então**
19: $s \leftarrow s'$
20: **fim se**
21: **até** atingir critério de parada

A hiperheurística Iterada possui uma fase de perturbação em que a estrutura de vizinhança a ser aplicada em uma solução corrente é selecionada de forma aleatória com distribuição de probabilidades iguais, dentre um conjunto de estruturas de vizinhanças disponíveis. Logo após, na fase de intensificação, heurísticas de baixo nível do tipo busca local são empregadas e a melhor solução encontrada é retornada. É feito uma comparação entre essa solução obtida e a melhor até então conhecida, se essa solução for melhor ela se torna a solução corrente, caso contrário é desprezada. A Figura 14 apresenta o funcionamento do método através do seu pseudocódigo.

Essa abordagem pode ser classificada como uma hiperheurística sem aprendizado de seleção de heurísticas de baixo de nível, já que há uma seleção de qual estrutura de

vizinhança usar em um determinado momento, além de não utilizar nenhum mecanismo de aprendizado que leve em consideração um histórico de busca.

Figura 15 – Hiperheurística baseada no Algoritmo Memético (Adaptado, MORENO, 2012).

Algoritmo HiperHeurísticaMemética()

1: $população \leftarrow CriaPopulaçãoInicial(10)$
2: **repete**
3: $s_1 \leftarrow TorneioBinário(população)$
4: $s_2 \leftarrow TorneioBinário(população)$
5: $h_1 \leftarrow HeurísticaDeCombinaçãoAleatória()$
6: $s' \leftarrow ExecutaHeurística(h_1, s_1, s_2)$
7: **se** $rand < 0,10$ **então**
8: $h_2 \leftarrow HeurísticaDeMutaçãoAleatória()$
9: $s' \leftarrow ExecutaHeurística(h_2, s')$
10: **fim se**
11: **se** $rand < 0,50$ **então**
12: $h_3 \leftarrow HeurísticaDeBuscaLocalAleatória()$
13: **senão**
14: $h_3 \leftarrow HeurísticaDeDestruiçãoERecriaçãoAleatória()$
15: **fim se**
16: $s' \leftarrow ExecutaHeurística(h_3, s')$
17: **se** $f(s_1) < f(s_2)$ **então**
18: $s_2 \leftarrow s'$
19: **senão**
20: $s_1 \leftarrow s'$
21: **fim se**
22: **até** atingir critério de parada

A hiperheurística baseada no Algoritmo Memético inicia com uma população de 10 soluções e a evolui, a cada iteração, trocando um indivíduo por um novo, o qual é obtido por uma recombinação de dois indivíduos escolhidos pelo torneio binário. Após o indivíduo é perturbado com uma probabilidade de 10% através de uma heurística de baixo nível de mutação. Finalmente esse indivíduo é modificado por uma fase de intensificação por meio de uma busca local ou ainda por uma heurística de destruição e recriação, tendo essas probabilidades iguais de serem escolhidas. O indivíduo produzido substitui o pior dos indivíduos selecionados pelo torneio binário. A Figura 15 mostra um pseudocódigo do algoritmo.

A abordagem também é classificada como uma hiperheurística de seleção de heurísticas sem aprendizado.

A hiperheurística Busca Tabu com aceitação adaptativa (TS - AA) foi baseada em uma proposta de Burke, Kendall e Soubeiga (2003). Nessa abordagem a seleção de heurísticas de baixo nível é feita através da atribuição de valores a cada método, os quais

Figura 16 – Hiperheurística Busca Tabu com Aceitação Adaptativa (TS - AA) (Adaptado, MORENO, 2012).

Algoritmo HiperHeurísticaBuscaTabu()

1: $s \leftarrow CriaSoluçãoInicial()$
2: Inicia o valor de cada heurística com 0
3: $\beta \leftarrow 0$
4: $T \leftarrow$ número de heurísticas - 1 (tabu *tenure*)
5: **repete**
6: $h \leftarrow HeurísticaDeMaiorValor()$
7: $s' \leftarrow ExecutaHeurística(h, s)$
8: **se** $f(s') < f(s)$ **então**
9: $AumentaValor(h, 1)$
10: **fim se**
11: **se** $f(s') > f(s)$ **então**
12: $EsvaziaListaTabu()$
13: $DecrementaValor(h, 1)$
14: $AdicionaNaListaTabu(h)$
15: **fim se**
16: **se** $f(s') = f(s)$ **então**
17: $AdicionaNaListaTabu(h)$
18: Remove heurísticas com mais de T iterações da lista tabu
19: **fim se**
20: **se** $f(s') < f(s)$ **ou** $rand < \beta$ **então**
21: $s \leftarrow s'$
22: **fim se**
23: **se** faz mais de 0,1 segundos desde o último progresso **então**
24: $\beta \leftarrow \beta + 0,05$
25: **fim se**
26: **se** faz mais de 0,1 segundos desde a última piora **então**
27: $\beta \leftarrow \beta - 0,05$
28: **fim se**
29: **até** atingir critério de parada

representam o desempenho da heurística. A cada iteração a heurística com o maior valor atribuído é escolhida, desde que essa não esteja na lista tabu. Os desempates são feitos com uma escolha aleatória com probabilidades iguais. O valor dado à um método é incrementado caso haja melhora na solução com sua aplicação e decrescido caso uma piora no valor da solução aconteça.

O critério para aceitar uma nova solução muda de acordo com o progresso da busca, em que todas as melhores soluções são aceitas, porém há certos momentos em que soluções piores podem ser aceitas visando a fuga de ótimos locais. A Figura 16 apresenta o pseudocódigo do algoritmo:

Essa abordagem pode ser classificada como uma hiperheurística de seleção de heurísticas com aprendizado online, já que há uma lista tabu interfere na seleção das heurísticas, sendo esse processo atualizado durante a busca.

4 MÉTODO PROPOSTO

4.1 MODELAGEM DO PROBLEMA DE APRENDIZADO

Um problema de AR precisa ser modelado matematicamente como um PDM e estar de acordo com todas as suas propriedades. O problema de AR em questão a ser solucionado pelo agente aprendiz é: indicar a heurística (construtiva) de baixo nível a ser aplicada na fase construtiva da metaheurística GRASP. Seja k um número definido de heurísticas de baixo nível e seja $H = \{h_1, h_2, h_3, \ldots, h_k\}$ um conjunto discreto contendo as heurísticas de baixo nível. O sistema aprendizagem do agente vai passar-se em um ambiente com matriz Q_{kxk}.

A modelagem a seguir tomou como base a modelagem proposta por Almeida (2014) para o Problema de Alocação de Facilidades. O problema pode ser modelado como um PDM por meio da quíntupla $M = \{T, S, A, R, P\}$ da seguinte forma:

- $T = \{1, 2, 3, \ldots, k\}$ o conjunto de instantes de decisão e k a cardinalidade de H.
- $S = \{s_1, s_2, s_3, \ldots, s_k\}$ o conjunto de estados, onde cada estado s_i equivale a uma $h_i \in H, i = 1, 2, 3, \ldots, k$.
- O conjunto de ações A é união dos conjuntos de ações disponíveis a cada estado s. Ou seja, $A = \{A(s_1) \cup A(s_2) \cup A(s_3) \cup \ldots \cup A(s_k)\}$. Uma ação corresponde no instante em que o processo muda de um estado atual $(s_i), \forall i \in 1, 2, 3, \ldots, k$ para outro (s_{i+1}) caso haja melhora no custo da solução gerada quando usada a heurística h representada pelo estado correspondente. Assume-se a restrição de que quando em um estado s, o processo não pode escolher uma ação que o faça continuar no mesmo estado.
- R é a função de recompensa $r(s, a)$, $\forall a \in A(s)$ e $s \in S$. O retorno atribui uma recompensa para as melhores ações. A recompensa é definida para esse problema como:

$$R = \frac{f(x^*)}{f(x) - f(x^*)}, \tag{4.1}$$

ou seja, a relação entre a melhor solução x^* encontrada até o momento e a diferença entre esta e a solução gerada no estado alcançado. Onde $f(x^*)$ é o custo da melhor solução obtida até a seguinte iteração e $f(x)$ o custo da solução obtida na fase construtiva utilizando a heurística h correspondente ao estado s_{i+1} alcançado pela ação a. Com essa equação admitimos que quanto mais próximo uma solução estiver

da melhor solução obtida, maior será a sua recompensa. Quando $f(x^*) = f(x)$ faz-se $R = f(x^*)/2$.

- P é a função de probabilidade de transição $p(s'|\ s, a)$, que consiste na probabilidade de se chegar à um estado s' estando no estado s escolhendo a ação a. Admite-se que $p(s'|\ s, a) = 1$, já que uma ação escolhida, por meio de um estado, sempre levará o processo de aprendizado para o estado selecionado pela ação.

4.2 POLÍTICA DE ATUAÇÃO DO ALGORITMO Q-LEARNING

A política de ações implementada pelo algoritmo *Q-learning* neste trabalho segue a seguinte estratégia. Em todos os estados, todas as ações estarão disponíveis, exceto aquela que faça o processo permanecer no estado corrente. Dado um estado atual s com uma heurística de baixo nível h associada e uma função $f(x)$ que retorna o custo da solução gerada aplicando-se essa heurística, a escolha da próxima ação a para a qual o processo seguirá, obedece os seguintes passos:

1. Seleciona-se uma ação a aleatoriamente a partir de $A(s)$ e aplica-se à fase construtiva do *GRASP* a heurística h associada ao estado retornado por essa ação;
2. Se a solução construída apresentar o custo menor que $f(x)$, o estado retornado pela ação selecionada será o próximo estado para o qual o processo irá seguir;
3. Caso contrário, retira-se a da lista de ações disponíveis e retorna-se ao passo 1.

4.3 MÉTODO GRASP DESENVOLVIDO

Fase construtiva

A fase de construção implementada utiliza como algoritmos para se obter uma solução factível ao PRVJT os dois algoritmos apresentados na Seção 3.1.2 (Heurísticas construtivas). O algoritmo Vizinho mais próximo orientado ao tempo para definir a ordem com qual os clientes devem ser inseridos na solução e o algoritmo *PFIH* para definir a rota e posição em que um cliente deverá ser inserido.

A implementação do parâmetro α para essa fase irá formar uma LRC que contém as melhores posições factíveis em ordem crescente (e suas respectivas rotas) para se inserir um cliente i definido pelo algoritmo de Vizinho mais próximo orientado ao tempo, ou seja, as posições com o qual caso o cliente i fosse inserido, o custo (distância total percorrida pelas rotas) da solução para o PRVJT seria minimizado. A LC contém todas as posições factíveis possíveis para as rotas já criadas.

A cada iteração da fase construtiva um cliente *i* ordenado pelo algoritmo de Vizinho mais próximo orientado ao tempo é inserido em uma rota definida pela escolha aleatória de uma posição feita da LRC. O método repete o procedimento obtenha uma solução factível e completa do PRVJT.

Pelo fato da heurística Vizinho mais próximo orientado ao tempo possuir diferentes configuração de seus parâmetros o método de construção a ser empregado dispõe desse conjunto de estratégias e a indefinição de qual ser utilizada é resolvida pelo método hiperheurístico, que terá no seu núcleo, o algoritmo *Q-learning*, que terá a terefa de indicar qual método de construção é o mais adequado para se obter melhores soluções.

Fase de busca local

A fase de busca local implementada utiliza como procedimentos de melhoria os dois algoritmos de refinamento descritos na Seção 3.1.2 (Heurísticas de busca local), os quais após uma solução obtida na fase anterior de construção buscam a partir desta solução encontrar uma solução de menor custo factível ao PRVJT explorando duas estruturas de vizinhança:

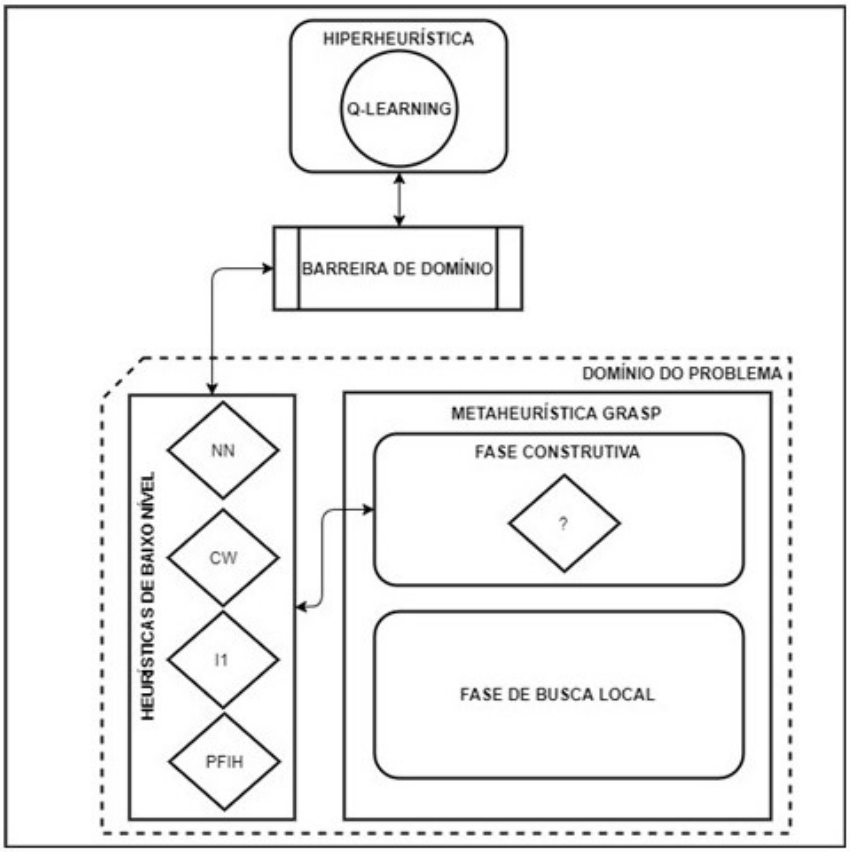
- Intra-rota: clientes envolvidos no movimento de troca fazem parte de uma mesma rota;
- Inter-rota: clientes envolvidos no movimento de troca fazem parte de rotas diferentes;

O movimento que explora as duas vizinhanças sucessivamente para encontrar melhores soluções é aquele troca de dois clientes de posição sejam esses de uma mesma rota ou de rotas diferentes. No algoritmo, o processo de busca local é executando respectivamente os movimentos da vizinhança Intra-rota e Inter-rota, até que não haja nenhuma melhoria na solução.

4.4 ESTRATÉGIA HIPERHEURÍSTICA PROPOSTA

O método proposto para resolver o PRVJT consiste em uma estratégia hiperheurística com aprendizado. A qual usa o conceito de hiperheurística *online* para implementar uma das fases da conhecida metaheurística *GRASP* utilizando uma técnica de aprendizado por reforço. Essa técnica é o algoritmo *Q-Learning*, o qual terá a tarefa de decidir qual método heurístico construtivo deve ser aplicada a cada iteração do método *GRASP* na primeira de suas duas fase que o compõe, a fase construtiva. A tarefa realizada pela técnica de aprendizado pode de certa forma indicar qual método construtivo pode

Figura 17 – Método proposto (Autoria própria).



ser mais eficiente quando utilizado em um problema específico. Já se pode haver uma quantidade razoável de configurações ou heurísticas construtivas diferentes na literatura que podem gerar uma solução para um dado problema, e atender o que a fase inicial do método *GRASP* necessita.

A eficiência do algoritmo é avaliada quanto a minimização da distância total percorridas pelos veículos empregados em uma solução para o PRVJT. Desse modo o algoritmo poderá no decorrer das iterações, indicar que heurística de baixo nível (construtiva) melhor se adapta a uma dada classe de instâncias ou a uma instância específica do problema. Podendo assim o método proposto ser melhor ajustado para resolver problema usando um método de construção mais eficiente. A Figura 17 representa um esquema do método desenvolvido.

Heurísticas de baixo nível

As heurísticas de baixo nível implementadas são baseadas nos métodos construtivos já apresentados na Seção 3.1.2 (Heurísticas construtivas). Essas estão disponíveis em 4 configurações diferentes e são definidas pelos dois métodos já descritos. Utilizam o algoritmo Vizinho mais próximo orientado a tempo para se obter a escolha do cliente

que deve ser atendido e o algoritmo *PFIH* para definir a rota e a ordem com qual o atendimento será feito.

As 4 heurísticas de baixo nível estão disponíveis em 4 configurações diferentes, ou seja, quatro funções de custo distintas para avaliar qual cliente deve ser escolhido para ser roteado. Tomando que os clientes são ordenados pela função de custo da heurística construtiva Vizinho mais próximo orientado a tempo, seguindo a equação 3.1, considerando o cliente i o depósito.

As quatro configurações que distingue as heurísticas de baixo nível são:

- $p_1 = 0,5; p_2 = 0,5; p_3 = 0,0$
- $p_1 = 0,0; p_2 = 1,0; p_3 = 0,0$
- $p_1 = 0,3; p_2 = 0,3; p_3 = 0,4$
- $p_1 = 0,4; p_2 = 0,4; p_3 = 0,2$

onde: $p_1(d_{ij}$:distância entre os nós i e $j)$, $p_2(t_{ij}$:diferença entre o serviço finalizado no cliente i e o atendimento do cliente $j)$ e $p_3(v_{ij}$:urgência do atendimento do cliente $j)$.

Barreira de domínio

A barreira de domínio implementada no método proposto usa de informações não ligadas diretamente ao problema para que o núcleo do método, ou seja, o algoritmo *Q-learning* possa inferir decisões de como deve ser gerenciado a escolha de um heurística construtiva que deve compor a fase de construção do método *GRASP*. O mecanismo de aprendizado usa das seguintes informações:

- Distância total percorrida pelos veículos empregados em uma solução;
- Número de veículos empregados em uma solução;
- Configurações de parâmetros das heurísticas de baixo nível.

4.5 HIPERHEURÍSTICAS IMPLEMENTADAS

Hiperheurística com aprendizado

O método com aprendizado proposto permite que a estratégia seja mais eficiente na escolha da heurística construtiva que possibilite encontrar soluções de boa qualidade, gerenciando de forma mais adequada o conjunto de métodos de construção disponíveis para a fase construtiva do *GRASP*. Possibilitando ainda que a segunda etapa do *GRASP*, a fase de busca local, seja acelerada.

A cada iteração o algoritmo, a escolha de uma heurística construtiva é baseada nas informações contidas na matriz dos *Q-valores* retornada pelo *Q-learning*. Essa escolha permite que testes bem sucedidos com um método sejam repetidos em iterações futuras.

A estratégia de escolha definida usa as seguintes etapas:

1. A matriz dos *Q-valores* é inicializada com valor 0 para todos os seus índices.

2. Depois, um índice da matriz dos *Q-valores* é sorteado de forma aleatória para iniciar o processo de atualização da mesma. Este índice passa a ser o estado s_0 para o algoritmo *Q-learning*;

3. A partir do estado s_0, usando a política descrita na Seção 4.2, o próximo estado s_1 é obtido;

4. Encontrados os estados s_0 e s_1, o processo iterativo que atualiza a matriz dos *Q-valores* é iniciado usando a equação (2.24). Depois de um determinado número de episódios, obtém-se a matriz *Q-valores* da qual será escolhido o método a ser usado como fase de construção. A escolha é feita escolhendo o índice que contém o maior valor da matriz *Q-valores*.

Depois de um número fixo iterações, a matriz *Q-valores* é atualizada pelo algoritmo *Q-learning*, que durante o processo de busca procura melhorar as informações contidas nesta. Os parâmetros de entrada do algoritmo são: o coeficiente de aprendizagem, o fator de desconto e o número de episódios.

O procedimento que indicará que método deve ser escolhido tem como entrada a matriz dos *Q-valores*. A técnica de aprendizado o algoritmo *Q-learning* tem como entrada o conjunto de heurísticas de baixo nível disponíveis; a melhor solução encontrada até o momento, para que seja caculada a recompensa que uma escolha irá receber, e os demais parâmetros de entradas da estratégia já citados. Logo após a execução da heurística construtiva escolhida como fase de construção do *GRASP*, a segunda fase, o procedimento de busca local é executado sem nenhuma alteração. O método retorna a melhor solução depois de um número de iterações máximas alcançadas definidas no *GRASP*. Esse método é denominado Hiperheurística-*Learning* (HH-L).

Hiperheurística aleatória

Esse método hiperheurístico não é dotado de nenhuma memória sobre o desempenho dos métodos construtivos disponíveis, nem de nenhuma técnica de aprendizado. A busca pela heurística construtiva que deverá fazer parte da fase de construção da metaheurística *GRASP* se dá de forma totalmente aleatória, ou seja, durante a execução do algoritmo, o método gerencia as heurísticas de baixo nível sem ter nenhuma informação da qualidade das soluções geradas por esses métodos.

No algoritmo, a cada determinado número de iterações, uma heurística de baixo nível é selecionada de forma aleatória para ser o método que irá constituir a fase construtiva da metaheurística *GRASP*. Logo após a fase de busca local é executado sem nenhuma alteração. O método retorna a melhor solução encontrada dentro de um número de iterações máximas definidos no *GRASP*. Esse método é denominada Hiperheurística-*Random* (HH-R).

5 RESULTADOS EXPERIMENTAIS

5.1 INSTÂNCIAS PARA O PRVJT

O método proposto e aqueles que irão ser usados nos comparativos de custo de solução entre os resultados obtidos utilizaram as conhecidas instâncias de teste propostas para o PRVJT de Solomon (1987). Essas 56 instâncias estão dispostas em três classes: *C*,*R* e *RC*.

Nos problemas do tipo *C* os clientes foram distribuídos em sub-regiões de forma agrupada. As instâncias da classe *R* os clientes estão dispostos de forma aleatória. E nos problemas da classe *RC*, os clientes são distribuídos usando das duas características das instâncias anteriores, ou seja, de maneira agrupada e aleatória.

Dentro de cada classe há ainda uma subdivisão em dois tipos, aquelas instâncias do tipo 1 ($C1, R1$ e $RC1$), os clientes são característicos por apresentarem janelas de tempo menores. Ao contrário daquelas do tipo 2 ($C2, R2$ e $RC2$), as quais os clientes dispõem de janelas de tempo com valores maiores. Essas características das instâncias permitem que uma rota da solução tenha um menor ou maior número de clientes.

As instâncias também se diferenciam entre a capacidade dos veículos têm para atender as demandas dos clientes. As instâncias $C1, R1$ e $RC1$ apresentam capacidade de 200 unidades. Já $C2, R2$ e $RC2$, as capacidades são de $700, 1000, 1000$ unidades respectivamente.

Os clientes presentes em cada instância variam nas quantidades de $25, 50$ e 100. Como informações de cada cliente ainda temos: as coordenadas da posição geográfica em relação ao depósito nos eixos abscissas e ordenadas, demanda, janelas de tempo no instante inicial e final e tempo de serviço.

As instâncias e os melhores resultados por métodos heurísticos estão disponíveis: https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/ os resultados ótimos foram obtidos por vários autores e são apresentados em Kallehauge et al. (2005).

5.2 EXPERIMENTOS COMPUTACIONAIS

5.2.1 Resultados dos experimentos computacionais

Os testes foram realizados em uma máquina com processador Intel Core i3 2.0 GHz, com 4 GB de memória RAM, em um sistema operacional Windows 10, arquitetura 64 bits.

Foram escolhidas 4 das 56 instâncias propostas por Solomon (1987), as quais apresentam diferentes características. Essas são: C102, C207, R101, RC102. A escolha das instâncias foi feita pensando em avaliar o desempenho do método em diversas situações que o problema propõe disponíveis nas instâncias de teste.

Os experimentos feitos usam o mesmo número de iterações *GRASP* (12) para cada método e as mesmas heurísticas e configurações disponíveis para essas. O parâmetro α foi ajustado empiricamente em 0,1. O algoritmo *Q-Learning* utilizado na Hiperheurística-*Learning* teve como valores para os parâmetros: coeficiente de aprendizagem (0,9), fator de desconto (1) e número de episódios definido como 5.

Tabela 1 – Valores da média da função objetivo para a HH-L e HH-R.

	HH-L			**HH-R**		
Instância	***Média***	***Tempo***	***σ(%)***	***Média***	***Tempo***	***σ(%)***
C102	1506,59	244,24	27,00	1549,35	100,73	64,39
C207	1206,31	436,02	44,70	1122,67	189,30	81,69
R101	2121,41	134,03	18,84	2125,82	72,73	20,96
RC102	2037,28	226,62	5,65	2040,47	103,19	8,17

Tendo em vista que os métodos implementados são algoritmos não determinísticos os valores apresentados na Tabela 1 são as médias de 30 execuções dos algoritmos para cada instância. Esses valores são referentes a média da função objetivo (em KM) e ao tempo de execução (em segundos), assim como também o desvio padrão para os valores de função objetivo. Como os algoritmo buscam minimizar a soma das distâncias percorrida pelos veículos empregados, o número de veículos utilizados foram desconsiderados para comparativos entre os métodos.

Pode-se observar que a Hiperheurística-*Learning* comparado a Hiperheurística-*Random* obteve melhores resultados em todas as instâncias para os valores médios das funções objetivo além de um desvio padrão menor. O que sugere que o mecanismo de aprendizado utilizado pôde indicar que método construtivo deveria ser empregado ao longo da execução da estratégia hiperheurística podendo gerar ótimos locais de melhor qualidade que aprimorados pela fase seguinte de busca local, possibilitou encontrar melhores soluções.

Com relação ao tempo de execução dos métodos a Hiperheurística-*Learning* teve um tempo de execução superior a Hiperheurística-*Random*. Isso se deve a técnica de AR que o método implementa.

A Tabela 2 apresenta a melhor, pior e média soluções encontradas por cada algoritmo para cada instância, assim como o desvio padrão desses valores. A melhores soluções encontradas pela Hiperheurística-*Learning* tem valores menores que aquelas obtidas pelo outro método em quase todas as instâncias testadas, com exceção da

Tabela 2 – Melhor, pior e média para os valores da função objetivo para a HH-L e HH-R.

	HH-L				**HH-R**			
Instância	***Melhor***	***Média***	***Pior***	***σ(%)***	***Melhor***	***Média***	***Pior***	***σ(%)***
C102	1482,35	1506,59	1710,35	125,22	1482,35	1549,35	1538,91	36,04
C207	1122,67	1206,31	1436,75	162,65	1100,79	1122,67	1281,99	98,90
R101	2106,62	2121,41	2157,02	25,90	2094,69	2125,82	2153,66	29,50
RC102	2033,74	2037,28	2064,37	16,75	2024,42	2040,47	2057,48	16,53

instância C102 onde os valores são iguais. O desvio padrão entre as soluções encontradas também é menor para Hiperheurística-*Learning*. O que indica que o método com aprendizado é capaz de gerar soluções mais próximas de sua média e que a escolha do método construtivo mais eficiente fez com que soluções melhores fossem encontradas.

Tabela 3 – Taxa de utilização (%) em que cada heurística de baixo nível foi método de construção de uma solução obtida para a HH-L e HH-R.

	HH-L				**HH-R**			
Heurística	***C102***	***C207***	***R101***	***RC102***	***C101***	***C207***	***R101***	***RC102***
H1	46,66	23,33	0	0	40	40	43,33	30
H2	40	40	100	6,66	26,66	16,66	20	20
H3	13,33	10	0	73,33	13,33	23,33	16,66	30
H4	0	26,66	0	20	20	20	20	20

Na Tabela 3 é apresentada em % o número de vezes em que cada heurística de baixo nível escolhida como método para compor a fase construtiva do *GRASP* encontrou a solução retornada pelo método. A variação entre esses valores sugere a dificuldade do algoritmo em inferir sobre qual método indicar, já que um método construtivo escolhido ainda está sujeito ao parâmetro α que permite gerar com diversidade soluções iniciais. Porém a diferença presente dos valores entre os algoritmos indica que o método com aprendizado foi capaz de tomar decisões favoráveis para encontrar melhores soluções a partir da escolha de um método construtivo mais eficiente.

A Figura 18 apresenta graficamente uma comparação entre os dois algoritmos implementados, levando em consideração os valores de função objetivo. Os dados mostrados nos gráficos foram submetidos a um processo de normalização, com o objetivo de ter a visualização do desempenho dos métodos melhorada. Esses valores foram normalizados pela média dos valores obtidos por cada hiperheurística, para cada uma das instâncias, de acordo com a equação:

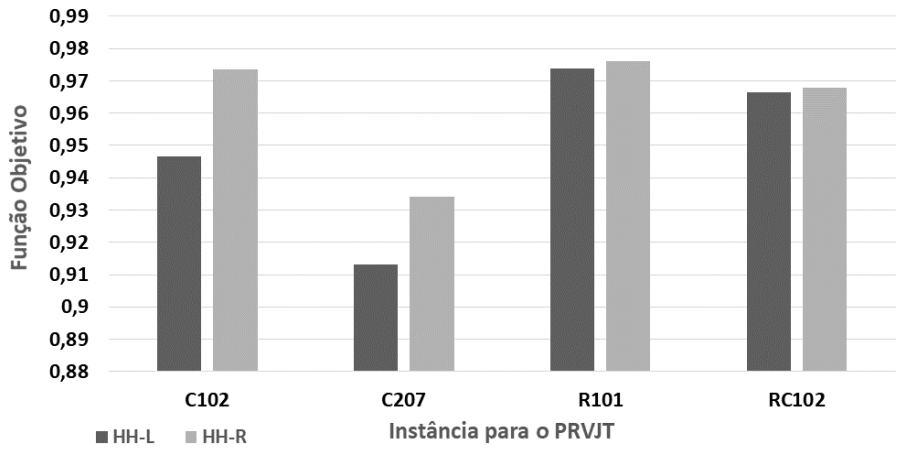
$$N_{ij} = V_{ij}/M_i, \forall i = 1, 2, 3, ..., n \tag{5.1}$$

onde:

$$M_i = \left(\sum_{j=1}^{m} V_{ij} \right) / m, \forall i = 1, 2, 3, ..., n \tag{5.2}$$

Na equação (5.1), considerando a instância i, N_{ij} é o valor normalizado executando um algoritmo j, V_{ij} representa os valores de função objetivo e tempo computacional executando o algoritmo j e M_i é a média dos valores. Na equação (5.2), m é o número de algoritmos usados e n a quantidade de instâncias testadas no experimento.

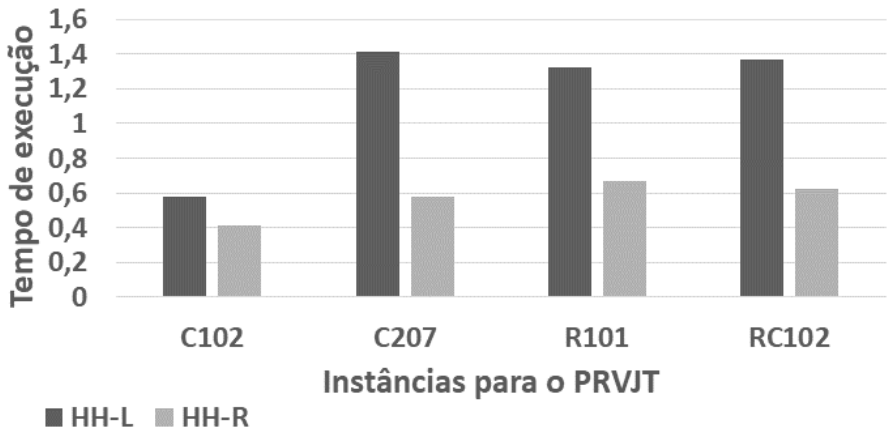
Figura 18 – Comparativo dos resultados da função objetivo para a HH-R e HHR-L.



Com o gráfico da Figura 18 percebemos que a Hiperheurística-*Learning* mostrou desempenho melhor que a Hiperheurística-*Random* em todas as instâncias testadas. Apresentado uma diferença de desempenho maior em relação as duas primeiras instâncias (C102, C207) pertencentes a classe que possui clientes dispostos de forma agrupada, onde as duas ainda tem configurações diferentes quanto as janelas de tempo dos clientes e capacidade do veículo designado para realizar o atendimento. Na instância R101, definida pela classe em que os clientes são distribuídos aleatoriamente e na instância RC102, em que os clientes estão dispostos usando as duas característica anteriores citadas o método com aprendizado apresentou desempenho mais semelhante ao outro, porém superior.

O gráfico apresentado na Figura 19 mostra que a Hiperheurística-*Learning* tem um tempo de execução superior a Hiperheurística-*Random* em todas as instâncias testadas. Isso se deve a presença da técnica de AR, onde o algoritmo *Q-learning* tem a necessidade de algumas iterações para realizar a tarefa de aprendizado, o que o outro método não precisa, já que a escolha do método construtivo é feita de forma aleatória.

Figura 19 – Comparativo do tempo de execução para a HH-R e HHR-L.



6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

CONCLUSÃO

A Hiperheurística-*Learning* apresentou desempenho superior ao outro método, a Hiperheurística-*Random*, quando consideramos os valores obtidos pela função objetivo de cada estratégia.

As diferença entre as duas estratégias é o motivo pela diferença de desempenho. Onde o emprego de uma técnica de AR, o algoritmo *Q-learning*, presente na Hiperheurística-*Learning*, tinha o objetivo de indicar dentro de um conjunto de métodos construtivos disponíveis a fazer parte da fase de construção da metaheurística *GRASP*, que método seria o mais indicado. Já se pode haver uma quantidade razoável de configurações ou heurísticas construtivas diferentes na literatura que podem gerar uma solução para um dado problema, e atender o que a fase inicial do método *GRASP* necessita.

No caso deste trabalho o problema abordado foi o Problema de Roteamento de Veículos com Janela de Tempo (PRVJT) e dois métodos construtivos foram combinados, a heurística de Vizinho mais Próximo e o método *Push-Foward Insertion Heurístic (PFIH)*, os quais geraram quatro heurísticas de baixo nível diferentes quanto as parametrizações. Essas quatro heurísticas de baixo nível foram os métodos em que o mecanismo de aprendizado teria de indicar qual seria o mais adequado a uma determinada instância. Já que essas poderiam ter desempenho diferentes não somente pelas suas configurações mas também pela presença do parâmetro α que permite gerar soluções diferentes de forma gulosa-aleatória.

Através do mecanismo de aprendizado o algoritmo foi capaz de usar o conhecimento adquirido em iterações anteriores para escolher a heurística construtiva que possa obter soluções promissoras. O aprendizado da técnica de AR foi implementado usando de informações presente na matriz dos Q-valores do algoritmo *Q-learning*.

A desvantagem da Hiperheurística com aprendizado ficou por conta do tempo de execução do algoritmo. Onde a técnica de AR precisou de mais tempo para realizar a tarefa de aprendizado enquanto a outra estratégia não possuía tal necessidade.

6.1 CONTRIBUIÇÕES DA DISSERTAÇÃO

O algoritmo Hiperheurística-*Learning* conseguiu oferecer soluções geradas a partir de um método que usasse de forma eficiente as heurísticas disponíveis, indicando aquela(s) mais adequada para uma dada instância. Onde a matriz dos Q-Valores do

algoritmo *Q-learning* foi atualizada inserindo a experiência adquirida pelo agente de aprendizado, onde foi possível usar de informações passadas em decisões futuras, aumentando a garantia de boas soluções e minimizando a tomada de decisões mais aleatórias.

Com isso, foi possível que as seguintes contribuições pudessem ser atingidas:

- Desenvolvimento de um método heperheurístico com aprendizado que pôde indicar dentro de um conjunto de métodos construtivos qual seria o mais adequado para constituir a fase de construção da metaheurística *GRASP* ao longo de sua execução.
- Obtenção de um algoritmo que utilizando AR que pode ser aplicado a problemas de otimização sem a necessidade de modelá-los como PDM;
- O método dotado do mecanismo de aprendizado obteve melhores soluções em qualidade comparado método que utilizava uma estratégia aleatória para buscar o método de construção mais adequado.

6.2 TRABALHOS FUTUROS

Como perspectivas de trabalhos futuros resultantes deste trabalho podemos considerar:

- Melhor parametrização do método *GRASP* e do algoritmo *Q-Learning* buscando reduzir o tempo de execução do algoritmo.
- Aplicação da hiperheurística com aprendizado implementando outras heurísticas de baixo nível, ou seja, diferentes métodos construtivos disponíveis na literatura para compor o método *GRASP* na fase construtiva.
- Aplicação de uma nova versão da hiperheurística com aprendizado, propondo uma diferente tarefa de aprendizado para o algoritmo *Q-learning*: indicar que heurística de refinamento da fase de busca local da metaheurística *GRASP* deve ser aplicada a uma classe de instância ou a uma instância específica do PRVJT.
- Aplicação da estratégia hiperheurística com aprendizado proposta a outras instâncias de grande porte do PRVJT disponíveis na literatura, assim como a outros problemas de Otimização Combinatória, de modo a verificar o comportamento e avaliar o desempenho do método proposto;
- Realizar uma análise estatística de soluções obtidas, em distância total percorrida e número de veículos empregados, de modo a constatar que a abordagem hiperheurística com aprendizado proposta contribui para encontrar melhores soluções que a estratégia hiperheurística construtiva aleatória.

REFERÊNCIAS

ALAZZAM, A.; LEWIS III, H. W. A new optimization algorithm for combinatorial problems. International Journal of Advanced Research in Artificial Intelligence (IJARAI), 2(5), 2013.

ALMEIDA, I. I. Metaheurística Híbrida Utilizando GRASP Reativo e Aprendizagem Por Reforço: Uma Aplicação na Segurança Pública. Dissertação de Mestrado – Universidade Estadual do Rio Grande do Norte – UERN, Mossoró, RN, 2014.

BADEAU, P.; GENDREAU, M.; GUERTIN, F.; POTVIN, J-Y.; TAILLARD, E. A parallel tabu search heuristic for the vehicle routing problem with time windows. Transportation Research, 5, p.109-122, 1997.

BAI, R.; KENDALL, G. An Investigation of Automated Planograms Using a Simulated Annealing Based Hyper-Heuristic. In: IBARAKI, T; NONOBE, K; (Ed). Metaheuristics: Progress as Real Problem Solvers. [S.1.]: Springer US, P.261-277, 2005.

BAKER, E. K. Vehicle routing with time windows constraints. Logistic and Transportation Review, 18(4), p.385-401, 1982.

BARBOSA, J. M. R. Aplicação de uma abordagem adaptativa de busca tabu a problemas de roteirização e programação de veículos. Dissertação (Mestrado) – Universidade Federal de São Carlos – UFSCar, SP, 2005.

BARD, J. F.; KONTORAVDIS, G.; YU, G. A Branch-and-Cut Procedure for the Vehicle Routing Problem with Time Windows, Transportation Science, 36(2), p.250-269, 2002.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surveys, n. 35, p. 268–308, 2003.

BODIN, L. D.; GOLDEN, B. Classification in Vehicle Routing and Scheduling. Networks, p. 97-108, 1981.

BODIN, L.; GOLDEN, B.; ASSAD, A. & BALL, M. Routing and Scheduling of Vehicles and Crews: The State of the Art. Computers & Operations Research, v. 10, p. 289-315, 1983.

BRÄYSY, O.; GENDREAU, M. Vehicle routing problem with time windows, part I: Route construction and local search algorithms. Transportation Science, 39, p.104-118, 2005.

BURKE, E. K.; BURKE, E. K.; CURTOIS, T.; HYDE, M.; KENDALL, G.; OCHOA, G.;

PETROVIC, S.; VÁSQUEZ-RODRIGUES, J. A. Hyflex: A Benchmark Framework for Crooss-domain Heuristic Search. arXiv.org, cs.Al, 2011.

BURKE, E. K.; CURTOIS, T.; HYDE, M.; KENDALL, G.; OCHOA, G.; PETROVIC, S.; VÁSQUEZ-RODRIGUES, J. A.; GRENDREAU, M. Iterated Local Search vs. Hyper-heuristics: Towards General-Purpose Search Algoritms. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE, Spain, Barcelona, p. 1-8, 2010b.

BURKE, E. K.; HYDE, M.; KENDALL, M.; OCHOA, G.; OZCAN, E.; WOODWARD, J. R. A Classification of Hyper-heuristic Approaches. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). Handbook of Metaheuristics. [S.1.]: Springer US, p. 449-468, 2010a.

BURKE, E.; KENDALL, G.; SOUBEIGA, E. A Tabu-Search Hyperheuristic for Timetabling and Rostering. Journal of Heuristics, 9(6):451–470, 2003.

CHAKHLEVITCH, K.; COWLING, P. Hyperheuristics: Recent Developments. In: COTA, C.; SEVAUX, M.; SÖRENSEN, K. (Eds). Adaptive and Multilevel Metaheuristics. [S.1.]: Springer Berlin/Heldelberg, p.3-29, 2008.

CHIANG, W. C.; RUSSELL, R. A. Simulated annealing metaheuristics for the vehicle routing problem with time windows. Ann.Operations Research, 63, p.3-27, 1996.

CHRISTOFIDES, N.; BEASLEY, J. The period routing problem, Networks 14, p. 237-246, 1984.

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, v. 52, p. 928–936, 2001.

COWLING, P.; KENDALL, G.; SOUBEIGA, E. A Hyperheuristic Approach to Scheduling a Sales Summit. In: Third International Conference on Practice and Theory of Automated Timetabling III. London, UK, p. 176-190, 2001.

CZECH, Z.; CZARNAS, P. A parallel simulated annealing for the vehicle routing problem with time windows. Proceedings of 10th Euromicro Workshop Parallel, Distributed Network-based Processing, p. 376-383, Canary Islands, Spain, 2002.

DANTZIG, G. B.; RAMSER, R. H. The Truck Dispatching Problem. Management Science, 6, p.80-91, 1959.

DANTZIG, G. B.; WOLFE, P. The decomposition algorithm for linear programming. Operations Research, 8, p.101-111, 1960.

DESROCHERS, M.; DESROSIERS, J. E.; SOLOMON, M. M. A new optimization algorithms for Vehicle Routing Problem with Time Windows. Operations Research, 40(2),

p.342-354, 1992.

DORIGO, M.; CARO, G. Ant Colony Optimization: A New Meta-Heuristic. Evolutionary Computation, v. 2, p.1470-1477, CEC 99, IEEE, 1999.

FANG, H.; ROSS, P.; CORNE, D.; A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems. In A. COHN, editor, Proceedings of ECAI 94: 11th European Conference on Artificial Intelligence, p. 590-594. Jonh Wiley and Sons, 1994.

FEO, T. A.; RESENDE, M. G. C. Greedy randomized adaptive search procedures. Journal of Global Optimization, v. 6, p. 109–133, 1995.

FESTA, P.; RESENDE, M. G. C. An annotated bibliography of GRASP - part i: Algorithm. In: [S.l : s.n.], 2009. p. 1–24.

FRAGA, M. C. P. Uma metodologia híbrida colônia de formigas – busca tabu – reconexão por caminhos para resolução do problema de roteamento de veículos com janelas de tempo. Dissertação de Mestrado em Modelagem Matemática e Computação, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2006.

FRANCO JÚNIOR, E. F.; CARVALHO, F. A; OLIVEIRA, H. C. B. Adaptação da Metaheurística GRASP na Resolução do Problema de Roteamento de Veículos com Janela de Tempo. Anais do III Encontro Regional de Pesquisa Operacional da Região Sul, p. 49-54, Porto Alegre, RS, 2009.

GAMBARDELLA, L. M.; TAILLARD, E.; AGAZZI, G. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, In New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover (eds), p.63-76, McGraw-Hill, London, 1999.

GENDREAU, M.; TARANTILIS, C. D. Solving large-scale vehicle routing problems with time windows: the state-of-the-art, Technical report CIRRELT-2010- 04, CIRRELT, Montreal, QC, Canadá, 2010.

GLOVER, F. Future paths for integer programming and links to artificialcintelligence. Computers & Operations Research, 13, p.533-549, 1986.

GOLDBARG, M. C.; LUNA, H. P. L. Otimização combinatória e programação linear: modelos e algoritmos, Campus, Rio de Janeiro, 2000.

GOLDBARG, M. C.; LUNA, Henrique L. P. Otimização combinatória e programação linear: modelos e algoritmos, 2 ed, Elsevier, Rio de Janeiro, 2005.

GRATCH, J.; CHEIN, S.; JONG, G. Learning Search Control Knowledge for Deep Space Network Scheduling. In Proceedings of the Tenth International Conference on Machine

Learning, p.135-142, 1993.

HOLLAND, J. H. Adaptation in Natural and Artificial System. Ann Arbor, Michigan: The University of Michigan Press, USA, 1975.

JEPSEN, M.; PETERSEN, B.; SPOORENDONK, S.; PISINGER, D. A non-robust branch and cutand-price algorithm for the vehicle routing problem with time windows. Technical Report, Department of Computer Science, University of Copenhagen, Denmark. 2006.

KALLEHAUGE, B.; LARSEN, J.; MADSEN, O. B. G.; SOLOMON, M. M. Vehicle routing problem with time windows. In: Desaulniers, G., Desrosiers, J., & Solomon, M. M. (Eds), Column Generation, GERAD 25th Anniversary Series, Springer, New York, 2005.

KIRKPATRICK, S.; GELLAT, C. D.; VECCHI, M. P. Optimization by simulated annealing. Science, 220, p.671-680, 1983.

KOLEN, A. W. J.; RINNOOY KAN, A. H. G.; TRIENEKENS, H. W. J. M. Vehicle routing with time windows. Operations Research, 35(2), p.266-273,1987.

KONTORAVDIS, G. A.; BARD, J. F. A GRASP for the vehicle routing problem with time windows. INFORMS J. on Computing, 7(1), p.10-23, 1995.

LARSEN, J. Parallelization of the vehicle routing problem with time windows, PhD Thesis, Departament of Mathematical Modeling Technical University of Denmark, 1999.

LAU, H. C.; SIM, M.; TEO, K. M. Vehicle routing problem with time windows and a limited number of vehicles. European Journal of Operational Research, 148, p.559-569, 2003.

LENSTRA, J. K.; RINNOOY KAN, A. H. G. Complexity of vehicle routing and scheduling problems. Networks, v. 11, n. 2, p. 221–227, 2006.

LI, H.; LIM, A. Local search annealing-like restarts to solve the VRPTW. European Journal of Operation Research 150, p. 115-127, 2003.

LIMA, J. C. Resolução do problema de roteamento de veículos com janela de tempo: uma abordagem através de algoritmos genéticos. Dissertação de Mestrado - Centro Federal de Educação Tecnológica de Minas Gerais - CEFET-MG, Belo Horizinte, MG, 2013.

LIMA JÚNIOR, F. C. Algoritmo Q-learning como Estratégia de Exploração e/ou Explotação para as Metaheurísticas GRASP e Algoritmo Genético. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte – UFRN, Natal, RN, 2009.

LIN, S. Computer solutions of the traveling salesman problem. J. Bell System Technical,

44, p.2245-2269, 1965.

MORENO, P. C. Uma abordagem Hiper-heurística inspirada em exames de partículas. Dissertação de Mestrado – Universidade de Fortaleza – UNIFOR, Fortaleza, CE, 2012.

OLIVEIRA, H. B. Um modelo híbrido estocástico para tratamento de um problema de roteamento de veículos com janela de tempo. Dissertação de Mestrado - CIN/UFPE, Recife, PE, 2007.

OMBUKI, B.; ROSS, B. J.; HANSHAR, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. Applied Intelligence, v. 1, n. 24, p. 17-30, 2006.

OSMAN, I. H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. Ann. Operations Research, 41, p.421-452, 1993.

OSMAN, I. H.; LAPORTE, G. Metaheuristics: A bibliography. Annals of Operations Research, v. 63, p. 513–623, 1996.

OZCAN, E.; UYAR, S. E.; BURKE, E. K. A greedy hyper-heuristic in dynamic enviroments. In: GEECO '09: Proceendings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers. Montreal, QC, Canadá: ACM, p.2201-2204, 2009.

POTVIN, J.-Y.; ROUSSEAU, J.-M. An exchange heuristic for routing problems with time windows. Journal of the Operations Research Society, 46, p.1433 – 1446, 1995.

PUTERMAN, M. L. Markov Decision Processes Discrete Stochastic Dynamic Programming. New York, USA: John Wiley e Sons, Inc, 2005.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptative search procedures. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). Handbook of Metaheuristics. [S.l.]: Kluwer Academic Publishers, p. 219–249, 2003.

ROSS, P.; BURKE, E.; HART, E.; KENDALL, G.; NEWALL, J.; SCHULENBURG, S. Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In: F. GLOVER and G. KOCHENBERGER, editors, Handbook of Metaheuristics, p.457–474. Kluwer, 2003.

SAVELSBERGH, M. W. P. The vehicle routing problem with time window: minimizing route duration, ORSA Journal on Computing 4, p.146-154, 1992.

SCHULZE, J.; FAHLE, T. A parallel algorithm for the vehicle routing problem with time window constraints. Annals of Operations Research, 86, p.585-607, 1999.

SOLOMON, M. M. Algorithms for the Vehicle Routing and Scheduling Problem with

Time Window Contraints, Operational Research, 35, p. 254-265, 1987.

SUCUPIRA, I. R. Um estudo empírico de hiper-heurísticas. Dissertação (Mestrado) — Instituto de Matemática e Estatísticas, Universidade de São Paulo, São Paulo. p.124, 2007.

SUTTON, R. S.; BARTO, A. G. Reinforcement Learning: An Introduction. MA: MIT Press, 1998.

STEHLING, T. M. Algoritmos Populacionais Híbridos aplicados ao Problema de Roteamento de Veículos com Janela de Tempo. Dissertação de Mestrado, Centro Federal de Educação Tecnológica de Minas, 2015.

TAILLARD, E.; BADEAU, P.; GENDREAU, M.; GEURTIN, F.; POTVIN, J. Y. A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows, Transportation Science, 31, p.170-186, 1997.

TAN, K. C.; LEE, L. H.; ZHU, Q. L.; OU, K. Heuristic methods for vehicle routing problem with time windows, Artificial Intelligence in Engineering, 15, p.281-295, 2001.

TANDABANI, A., PRIYAN, K., JANAKIRAMAN, S., E POTHULA, S. A comparative study of metaheuristic approach for cutting stock problem. In: 2016 International Conference on Communication and Electronics Systems (ICCES), p. 1-4, 2016.

TERASHIMA-MARÍN, H.; ROSS, P.; VALENZUELA-RENDÓN, M. Evolution of Constraint Satisfaction Strategies in Examination Timetabling. In: BANZHAF, W. et al., editors, Proceedings of the GECCO-99 Genetic and Evolutionary Computation Conference, Morgan Kaufmann, p. 635- 642, 1999.

THANGIAH, S. R.; OSMAN, I. H.; SUN, T. Hybrid Genetic Algorithm Simulated Annealing and Tabu Search Methods for Vehicle Routing Problem with Time Windows. Technical Report 27, Computer Science Department, Slippery Rock University, 1994.

TOTH, P.; VIGO, D. The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, U.S.A, 2002.

VIEIRA, H. P. Metaheurística para a solução de problemas de roteamento de veículos com janela de tempo. Dissertação de Mestrado – Universidade Estadual de Campinas – UNICAMP, São Paulo, SP, 2008.

WATKINS, C. J. C. H. Learning from Delayed Rewards. Tese (Doutorado) — University of Cambridge, England, 1989.

WOCH, M.; LEBKOWSKI, P. Sequential Simulated Annealing for the Vehicle Routing Problem with Time Windows. Decision Making in Manufacturing and Services, vol. 3,

No 1-2. P.87-100, 2009.

APÊNDICE

Neste apêndice contém a listagem das soluções obtidas pelos algoritmos em cada execução para as instâncias de teste. Onde NV - número de veículos empregados, DT - distância total percorrida (Função objetivo), TE - tempo de execução e HE - heurística utilizada na fase construtiva no *GRASP* para obter a solução.

Tabela 4 – Instância - C102

	HH-R				**HH-L**			
EXECUÇÃO	NV	DT	TE	HE	NV	DT	TE	HE
1	10	1535,55	96,83	1	10	1482,35	234,17	0
2	10	1593,41	104,09	2	10	1485,28	247,83	0
3	10	1701,69	111,6	3	10	1482,35	239,72	0
4	10	1482,35	93,32	0	10	1535,55	223,31	2
5	10	1482,35	114,27	0	10	1482,35	240,59	0
6	10	1666,7	105,1	3	10	1538,34	237,83	1
7	10	1538,91	113,26	0	10	1482,35	230,78	0
8	10	1522,65	93,14	2	10	1538,34	239,68	2
9	10	1623,77	96,16	3	10	1538,34	235,2	2
10	10	1482,79	112,79	1	10	1482,35	234,4	1
11	10	1710,35	99,22	3	10	1482,35	226,41	0
12	10	1535,55	96,36	0	10	1528,83	223,05	1
13	10	1538,34	109,76	0	10	1490,02	241,03	1
14	10	1535,55	97,46	1	10	1483,66	238,78	1
15	10	1482,79	96,9	0	10	1538,34	255,85	0
16	10	1482,35	100,2	1	10	1538,91	249,25	1
17	10	1538,91	87,31	0	10	1490,02	242,91	0
18	10	1538,34	87,12	1	10	1538,34	243,49	1
19	10	1535,55	124,4	0	10	1482,35	252,72	0
20	10	1482,35	111,69	0	10	1538,91	248,66	1
21	10	1490,02	90,76	0	10	1538,34	237,16	1
22	10	1522,65	99,14	3	10	1482,35	247,22	0
23	10	1483,66	96,17	1	10	1482,35	215,93	0
24	10	1535,55	102,34	2	10	1482,35	297,13	1
25	10	1629,16	91,74	1	10	1535,55	258,51	2
26	10	1610,69	96,1	0	10	1482,35	266,05	0
27	10	1535,55	95,96	0	10	1531,56	257,07	1
29	10	1535,55	97,45	1	10	1482,35	250,88	0
29	10	1590,89	98,88	2	10	1538,91	249,94	1
30	10	1536,75	102,42	3	10	1482,35	261,9	0
MÉDIA		1549,35	100,73			1506,59	244,24	
DESVIO PADRÃO		64,39				27,00		

Tabela 5 – Instância - C207

	HH-R				**HH-L**			
EXECUÇÃO	NV	DT	TE	HE	NV	DT	TE	HE
1	4	1176,95	160,54	0	4	1163,96	422,85	1
2	4	1171,69	180,27	3	4	1176,95	439,14	2
3	4	1204,48	167,45	3	4	1170,96	398,12	1
4	4	1186,83	165,24	0	4	1200,6	451,87	2
5	4	1263,22	180,05	0	4	1236,99	441,79	3
6	4	1253,76	188,89	1	4	1232,07	465,01	2
7	4	1219,66	186,24	1	4	1160,41	479,94	0
8	4	1143,78	142,278	2	4	1175,23	456,13	0
9	4	1150,02	183,95	0	4	1273,34	457,09	1
10	4	1189,4	160,2	0	4	1205,87	434,42	1
11	4	1347,1	192,15	0	4	1213,91	421,39	0
12	4	1243,19	184,18	0	4	1226,79	449,69	1
13	4	1127,36	182,98	0	4	1183,42	444,51	1
14	4	1250,41	207,3	3	4	1200,96	401,11	0
15	4	1277,44	172,8	0	4	1187,91	443,6	1
16	4	1343,9	214,96	3	4	1231,5	405,94	1
17	4	1301,77	184,61	2	4	1197,2	447,08	1
18	4	1436,75	171,34	1	4	1224,49	427,84	1
19	4	1226,13	169,97	1	4	1279,39	437,79	0
20	4	1192,84	186,51	3	4	1267,93	422,18	3
21	4	1207,75	178,99	2	4	1183,42	407,22	0
22	4	1277,44	147,92	2	4	1198,35	456,13	1
23	4	1157,11	150,14	2	4	1216,46	444,1	3
24	4	1158,83	181,95	0	4	1192,91	450,1	0
25	4	1243,26	179,7	0	4	1232,16	450,55	3
26	4	1166,87	198,26	3	4	1182,86	467,7	1
27	4	1354,96	190,61	1	4	1281,99	428,18	3
29	4	1413,77	230,96	2	4	1100,79	390	3
29	4	1208,44	192,47	2	4	1111,39	417,73	3
30	4	1122,67	189,3	0	4	1279,36	421,68	3
MÉDIA		1233,92	180,74			1206,31	436,02	
DESVIO PADRÃO		81,69				44,70		

Tabela 6 – Instância - R101

	HH-R				**HH-L**			
EXECUÇÃO	NV	DT	TE	HE	NV	DT	TE	HE
1	24	2146,55	84,71	0	24	2106,62	138,77	1
2	24	2123,48	90,34	0	24	2106,62	130,38	1
3	24	2128,25	69,9	3	24	2106,62	136,66	1
4	24	2106,62	66,97	0	24	2146,55	137,57	1
5	25	2157,02	61,84	0	24	2106,62	131,22	1
6	23	2125,07	80,36	0	24	2146,8	129,62	1
7	23	2125,07	84,22	0	24	2106,62	130,99	1
8	25	2157,02	64,19	2	24	2106,62	134,84	1
9	24	2123,48	72,17	0	24	2153,66	134,52	1
10	24	2106,62	77,81	2	24	2106,62	131,45	1
11	24	2146,55	68,9	0	24	2123,48	137,36	1
12	25	2157,02	60,19	1	24	2149,49	139	1
13	23	2128,25	61,86	3	24	2106,62	136,11	1
14	24	2106,62	63,68	1	24	2123,48	132,72	1
15	24	2106,62	64,96	2	24	2094,69	133,09	1
16	24	2106,62	68,07	3	24	2123,48	139	1
17	24	2106,62	61,65	1	24	2146,55	133,07	1
18	24	2106,62	65,15	1	24	2106,62	138,1	1
19	24	2106,62	65,65	3	24	2106,62	133,37	1
20	25	2157,02	61,42	2	24	2106,62	133,64	1
21	24	2146,55	64,58	0	24	2106,62	137,93	1
22	24	2157,02	63,48	0	24	2123,48	124,7	1
23	24	2106,62	65,72	1	24	2146,55	134,73	1
24	24	2106,62	61,78	1	24	2146,8	124,21	1
25	24	2157,02	64,11	0	24	2106,62	131,01	1
26	24	2106,62	67,58	0	24	2146,55	131,42	1
27	24	2106,62	65,84	3	24	2106,62	135,83	1
29	24	2106,62	64,86	3	24	2106,62	134,85	1
29	24	2146,55	64,12	2	24	2146,55	139,81	1
30	24	2106,62	72,73	0	24	2125,07	135,01	1
MÉDIA		2125,82	68,29			2121,41	134,03	
DESVIO PADRÃO		20,96				18,84		

Tabela 7 – Instância - RC102

	HH-R				**HH-L**			
EXECUÇÃO	NV	DT	TE	HE	NV	DT	TE	HE
1	18	2056,5	94,19	1	18	2036,05	238,77	2
2	18	2037,89	83,92	1	18	2040,37	245,59	2
3	18	2036,05	92,01	0	18	2033,74	233,76	2
4	18	2040,57	91,19	2	18	2035,88	239,14	3
5	18	2036,05	93,57	1	18	2040,57	217,86	2
6	18	2040,57	98,95	0	18	2024,42	222,27	1
7	18	2061,4	92,02	0	18	2033,74	231,82	2
8	18	2040,57	121,11	2	18	2033,74	238,57	3
9	18	2036,05	101,07	2	18	2036,05	242,32	2
10	18	2064,37	100,25	3	18	2031,2	212,31	3
11	18	2040,57	100,96	3	18	2057,48	209,4	1
12	18	2040,57	118,36	0	18	2033,74	246,48	2
13	18	2037,89	118,41	0	18	2036,08	274,35	2
14	18	2034,01	108,34	1	18	2036,05	230,38	2
15	18	2039,57	99,33	0	18	2036,05	224,4	2
16	18	2033,74	119,24	2	18	2037,89	198,78	2
17	18	2040,57	104,27	0	18	2041,61	202,82	2
18	18	2035,84	98,72	0	18	2036,05	181,39	2
19	18	2036,05	105,98	0	18	2033,74	187,79	2
20	18	2036,05	91,64	3	18	2036,05	222,81	2
21	18	2036,05	123,64	3	18	2036,05	235,95	2
22	18	2036,05	106,04	2	18	2034,06	224,32	2
23	18	2033,74	107,69	2	18	2036,05	236,13	3
24	18	2056,49	119,96	2	18	2036,05	237,28	2
25	18	2036,05	90,91	3	18	2048,11	221,2	3
26	18	2041,61	83,83	3	18	2040,57	239,09	2
27	18	2043,11	96,25	1	18	2040,57	222,96	2
29	18	2034,01	102,49	2	18	2043,11	216,26	3
29	18	2036,05	126,5	1	18	2034,01	222,84	2
30	18	2036,05	104,92	2	18	2039,57	241,84	2
MÉDIA		2440,47	103,19			2037,28	226,72	
DESVIO PADRÃO		8,17				5,65		