Universidade Federal do Maranhão - UFMA
Centro de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação

Resolvendo um problema de roteamento de veículos capacitado por meio de uma meta-heurística GRASP: um estudo de caso

Mayrla Santos Jansen

São Luís - MA

Fevereiro de 2022

Mayrla Santos Jansen

Resolvendo um problema de roteamento de veículos capacitado por meio de uma meta-heurística GRASP: um estudo de caso

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Universidade Federal do Maranhão – UFMA

Centro de Ciências Exatas e Tecnológicas

Curso de Ciência da Computação

Orientador: Francisco Glaubos Nunes Clímaco

São Luís - MA

Fevereiro de 2022

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Santos Jansen, Mayrla.
Resolvendo um problema de roteamento de veículos capacitado por meio de uma meta-heurística GRASP: um estudo de caso / Mayrla Santos Jansen. - 2022.
38 p.

Orientador(a): Francisco Glaubos Nunes Clímaco.
Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2022.

1. GRASP. 2. Pesquisa Operacional. 3. Problema de Roteamento de Veículos Capacitados. I. Glaubos Nunes Clímaco, Francisco. II. Título.

Mayrla Santos Jansen

Resolvendo um problema de roteamento de veículos capacitado por meio de uma meta-heurística GRASP: um estudo de caso

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís - MA, 03 de fevereiro de 2022:

Francisco Glaubos Nunes Clímaco
Orientador

Carlos Eduardo Portela Serra de Castro
Convidado

Mário Antônio Meireles Teixeira
Convidado

São Luís - MA
Fevereiro de 2022

Agradecimentos

Em primeiro lugar, agradeço a Deus por sempre iluminar e abençoar meu caminho até este momento se tornar real e nunca me deixar faltar fé e força.

Agradeço a minha mãe que batalhou e lutou junto comigo, se tornou minha maior fã e em todos os momentos se mostrou o meu porto seguro, minha motivação e lugar de amor.

Agradeço ao meu pai, por todos os ensinamentos, confiança e certeza depositada em mim e no meu sonho. E ao meu avô, que fez sempre do meu sonho o seu próprio.

Agradeço às minhas melhores amigas, Laryssa e Wanessa, com suas famílias, por todo o apoio, carinho e força de vontade para me ajudar e me levar a seguir quando precisei, por sempre tornarem a jornada mais fácil.

A todos os amigos e familiares, que sempre se colocaram a disposição de me ajudar e amparar quando necessitei. Que me mostraram muitas vezes o melhor caminho e deram leveza nos momentos mais pesados.

Agradeço ao professor Francisco Glaubos por aceitar comigo esse desafio, pela paciência e força de sempre buscar o meu melhor para este trabalho.

Obrigada a todos de coração!

"Computação não se relaciona mais a computadores.
Relaciona-se a viver...."
(Nicholas Negroponte)

Resumo

Na área de Pesquisa Operacional um problema muito estudado é o Problema de Roteamento de Veículos (PRV). Uma extensão desse problema é o Problema de Roteamento de Veículos Capacitados (PRVC) definido como a busca para determinar um conjunto de rotas para uma frota homogênea de veículos, a partir de um depósito central com destino a um conjunto de clientes que demandam determinado produto. Este trabalho propõe uma método de solução para o PRVC por meio da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP). Para validar o método proposto, foram criadas instâncias (cenários) do problema baseadas em dados reais fornecidos pela empresa Loggi. Além disso, é realizada uma comparação entre os resultados obtidos com a meta-heurística GRASP e o uso de uma busca gulosa para a resolução desse problema.

Palavras-chaves: Pesquisa Operacional, Problema de Roteamento de Veículos, Problema de Roteamento de Veículos Capacitados, GRASP.

Abstract

In the field of Operations Research, a problem that has been studied a lot is the Vehicle Routing Problem (PRV). An extension of this problem is the Qualified Vehicle Routing Problem (PRVC) defined as the search to determine a set of routes for a homogeneous fleet of vehicles, from a central warehouse destined for a set of customers who demand a certain product. This work proposes a solution method for PRVC using the *Greedy Randomized Adaptive Search Procedure* (GRASP) metaheuristic. To validate the proposed method, instances (scenarios) of the problem were created based on real data provided by the company Loggi. Furthermore, a comparison is made between the results obtained with the GRASP meta-heuristic and the use of a Greedy Search to solve this problem.

Keywords: Operations Research, Vehicle Routing Problem, Capacitated Vehicle Routing Problem, GRASP.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AG	Algoritmo Genético
API	Application Programming Interface
BATA	Backtracking Adaptive Threshold Accept
GIS	Sistema de Informações Geográficas
GRASP	Greedy Randomized Adaptive Search Procedure
JSON	JavaScript Object Notation
PCV	Problema do Caixeiro Viajante
PRV	Problema de Roteamento de Veículos
PRVC	Problema de Roteamento de Veículos Capacitados
RDBMS	Sistema de Gerenciamento de Banco de Dados Relacional
SDSS	Spatial Decision Support Systems
TS	Tabu Search

Sumário

1 Introdução

Ao lidar com a venda de um produto, requisitos de logística são aplicados de maneira a incluir todo o conjunto de operações, que se inicia desde o anúncio do produto à entrega nas mãos do consumidor. O transporte e entrega de produtos com agilidade e facilidade para o consumidor são serviços que se tornaram imprescindíveis na sociedade atual (SANTOS; SANTOS; BERTO, 2009).

Diariamente, empreendedores buscam melhorar a agilidade no quesito de transporte de mercadorias, a fim de conseguir uma vantagem no mercado, dado o alto nível de competitividade. Com o avanço tecnológico, o poder e a facilidade de compra ficou maior e atrelado a isso, o consumidor deseja receber o produto com mais agilidade. Além disso, a compra pela internet (influenciada pelas redes sociais) e a facilidade que o consumidor encontrou em não precisar se deslocar para efetuar uma compra (MENEGATTI et al., 2017), aumentou a demanda por bens e serviços.

Este trabalho investiga um estudo de caso do Problema de Roteamento de Veículos Capacitado (PRVC), que está relacionado diretamente com operações logísticas de entrega de produtos. O PRVC consiste em encontrar o menor custo de roteiro (conjunto de rotas) utilizando veículos semelhantes para atender demandas de clientes (por ex. entrega de produtos).

Em seu trabalho, (SANTOS, 2014) apresenta a viabilidade da aplicação de métodos de roteirização, considerando a grande representatividade do modal rodoviário na matriz de transporte brasileira, e os benefícios observados pela sua utilização. Para isso, foi utilizado um modelo matemático, que foi testado a partir de instâncias disponíveis na literatura e de um cenário hipotético baseado em municípios da região de Formiga-MG.

Além disso, (AKHTAR et al., 2017) apresenta uma ferramenta viável para otimizar rotas de coleta de resíduos para reduzir custos econômicos e impactos ambientais, utilizando o PRVC com o conceito de lixeira inteligente para encontrar as melhores soluções otimizadas de rota de coleta de resíduos. Os resultados obtidos mostram uma redução de 36,80% da distância da coleta total de resíduos, o que acaba aumentando a eficiência média de coleta de resíduos em 36,78% e reduz o consumo de combustível, custo de combustível e emissão de $CO2$ em 50%, 47,77%. e 44,68% respectivamente.

1.1 Motivação

As despesas com as operações devem ser cada vez mais pensadas do ponto de vista logístico, para que o produto final obtenha o menor valor de custo possível. Dessa

forma, a utilização de Pesquisa Operacional para este fim, se torna de grande valia, visto que a aplicação de suas técnicas buscam o máximo desempenho operacional, vantagens estratégicas e contabiliza o impacto financeiro (ARAÚJO; LIMA; LIMA, 2018).

No contexto da empresa Loggi, que é uma empresa de entregas, a otimização de rotas é uma boa prática a ser utilizada (GOMES et al., 2019), visto que uso de *softwares* de otimização podem proporcionar economia de 5% para uma empresa (HASLE; LIE; QUAK, 2007), já que o transporte é geralmente um componente significativo do custo de um produto (aprox. 10%) (RODRIGUE; COMTOIS; SLACK, 2016).

Para otimizar as rotas, podemos fazer uma relação com o Problema de Roteamento de Veículos (PRV), que consiste em determinar as melhores rotas para uma frota de veículos, de forma que todos os pontos (clientes, cidades, etc.) sejam visitados ao menor custo possível. O Problema de Roteamento de Veículos e sua extensão, o PRVC, são derivados do Problema do Caixeiro Viajante (PCV) tal que os primeiros relatos são datados de 1800 (GUIMARÃES; PEREIRA; MEDEIROS, 2012).

Por exemplo, é possível utilizar uma modelagem do PRV para otimizar a logística de uma empresa de entregas de cartas. Todos os produtos a serem entregues estão em um mesmo ponto (depósito) e é necessário que utilizem o melhor caminho para que as cartas sejam distribuídas em diversos pontos, como mostra a Figura 1. Um exemplo de solução é mostrado na Figura 2 onde são realizadas três rotas (vermelha, azul e verde) para a distribuição das cartas.

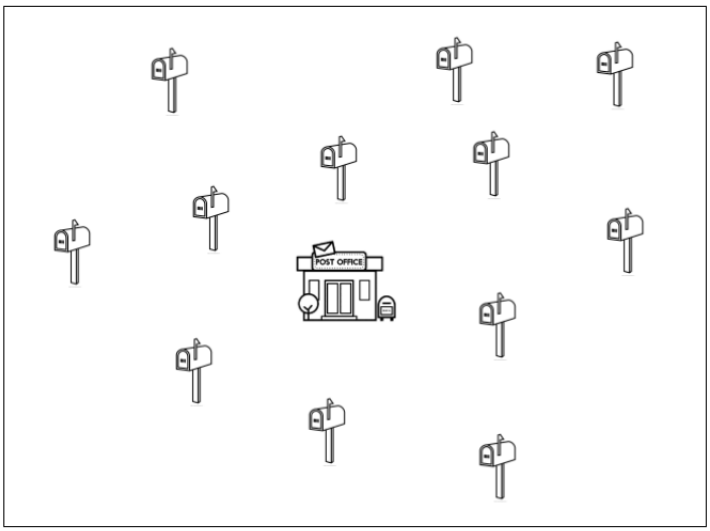

Figura 1 – Exemplo de PRV com entrega de cartas. Fonte: Própria do autor.

A otimização para a empresa Loggi é similar a do exemplo, contudo, sabemos que se utilizam veículos homogêneos para a distribuição. Sendo assim, este estudo fará uso do

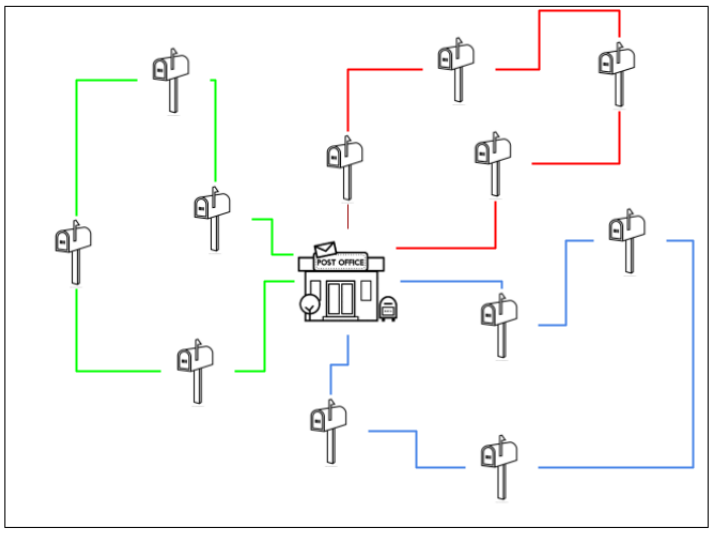

Figura 2 – Exemplo de solução para o PRV de entrega de cartas. Fonte: Própria do autor.

PRVC, visto que melhor se enquadra nos objetivos da roteirização para a empresa Loggi: entregas com menor custo e utilizando veículos homogêneos.

1.2 Objetivo

O objetivo principal deste trabalho é apresentar o desenvolvimento de uma heurística GRASP para a resolução do problema de roteamento de veículos capacitado, utilizando a meta-heurística GRASP. Como objetivos específicos, podemos destacar o estudo de caso com dados da empresa Loggi, a criação de instâncias e cenários de simulação, a partir de uma base de dados disponibilizada pela empresa Loggi; a contrução do algoritmo GRASP e a calibração de seus parâmetros.

1.3 Organização do trabalho

O trabalho está divido em cinco capítulos, sendo que neste capítulo foi apresentado a contextualização do problema, como também o objetivo deste trabalho. No Capítulo 2 será apresentada a fundamentação teórica desse trabalho: o PRVC, as heurísticas e em especifico a meta-heurística utilizada para realização deste trabalho. A metodologia trabalhada e todas as suas particularidades serão apresentados no Capítulo 3. Os resultados computacionais encontrados serão apresentados no Capítulo 4. Por fim, o Capítulo 5 destacará as principais conclusões obtidas e, ainda, como este trabalho pode ser continuado.

2 Fundamentação Teórica

2.1 Problema de Roteamento de Veículos Capacitado

O Problema do Roteamento de Veículos (PRV) busca definir um conjunto de rotas entre um ponto inicial em comum (depósito) e um conjunto de pontos que tenham uma certa demanda, com o objetivo de minimizar a quantidade de veículos, a distância percorrida e/ou o custo de tempo (ASSIS, 2007). Já o PRVC é a versão capacitada do PRV, tal que a diferença se encontra no fato de que agora os veículos são homogêneos e a demanda passa a ser objeto de grande valor (BITTENCOURT et al., 2012).

A Figura 3 representa um exemplo de solução para o PRV, onde tem-se quatro rotas distintas saindo de um depósito e que contempla todos os clientes necessários. Contudo, não se aplica ao PRVC pois não leva em consideração as demandas de cada cliente para a elaboração das rotas.

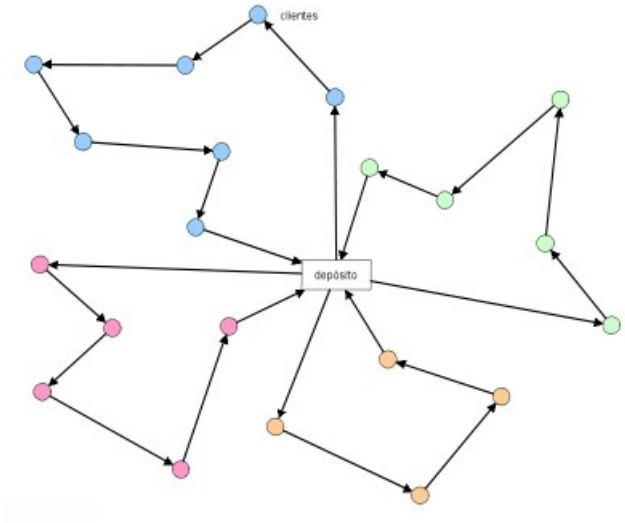

Figura 3 – Instância do PRV. Fonte: Logística Descomplicada.

2.1.1 História

O estudo do Problema de Roteamento de Veículos foi proposto inicialmente por Dantzig e Ramser (1959) para a resolução de um problema de distribuição de combustíveis em uma cidade. Por se tratar de uma derivação do Problema do Caixeiro Viajante também é classificado como um problema NP-difícil (problemas tão difíceis quanto os problemas mais difíceis) (SOUZA, 2013).

Para definir o PRVC, pode-se considerar um grafo (rede) com n nós, tal que, para cada par de nós (i, j) há um custo associado, que pode ser a distância entre os nós, o tempo levado para percorrer de um nó ao outro, o combustível usado no percurso, entre outros. Neste estudo de caso, o custo associado ao par de nós é dado pela distância em quilômetros de um nó ao outro. Além disso, para cada nó j existe um valor de demanda associado, e as seguintes restrições foram consideradas:

- Cada rota deve iniciar e terminar em um mesmo ponto (depósito);
- Cada cliente deve ser visitado apenas uma vez e somente por um único veículo, ou seja, o veículo deve ter capacidade para alocar toda a demanda de um cliente;
- A soma das demandas dos clientes incluídos em uma rota não deve exceder a capacidade do veículo.

Dessa maneira, o objetivo do problema é definido como encontrar um conjunto de rotas, sendo que cada uma delas deve ser percorrida por um veículo, enquanto minimiza o custo total do roteiro (conjunto de rotas). A Figura 4 ilustra uma possível solução para uma instância do PRVC, com uma frota de três veículos homogêneos (mesma capacidade) e oito cidades a serem visitadas.

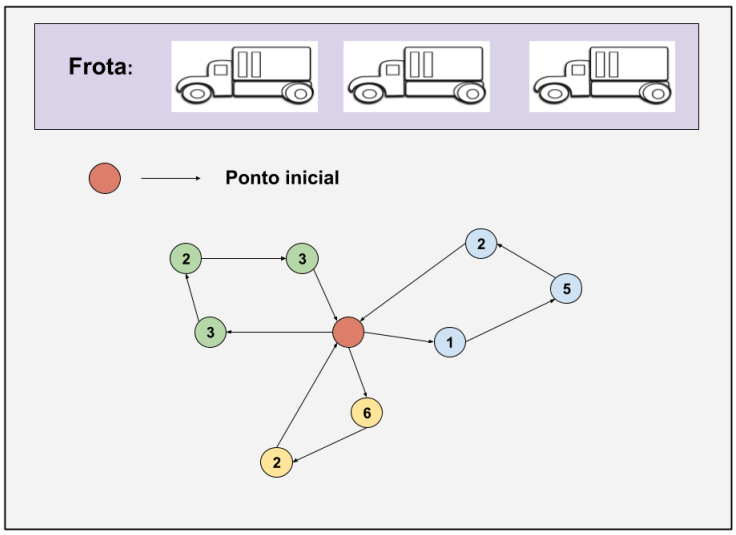

Figura 4 – Exemplo de solução para PRVC. Fonte: Própria do autor.

2.2 Trabalhos relacionados

A heurística mais famosa relacionada a resolver o PRVC é o algoritmo construtivo de (CLARKE; WRIGHT, 1964) (CWS). Esse algoritmo lida com o problema de roteamento

de uma frota de caminhões de capacidades variadas, e de um depósito central para vários pontos de entrega que pode exigir uma seleção de um número muito grande de rotas possíveis, se o número de pontos de entrega também for grande. Considerando alguns aspectos teóricos do problema, desenvolve um procedimento iterativo que permite a rápida seleção de uma rota ótima ou quase ótima. Essa heurística foi programada originalmente para um computador digital, mas também é adequada para computação manual.

Com o intuito de aprimorar o algoritmo CWS, foram propostas muitas variantes. Duas importantes variações são a de (MOLE; JAMESON, 1976) e de (HOLMES; PARKER, 1976). A primeira utiliza métodos de construção de rotas sequenciais e concorrentes para o planejamento de múltiplas jornadas de entrega, e conclui-se que o primeiro ofereceria vantagens práticas importantes se a qualidade das rotas pudesse ser melhorada. O critério de economia de Clarke e Wright é generalizado e cada viagem é automaticamente livre de cruzamentos. O método também é computacionalmente eficiente como evidenciado por um estudo da sensibilidade em relação ao tamanho da frota, de mudanças na capacidade do veículo e uma restrição de distância percorrida para um problema de depósito único com 225 clientes.

A segunda, introduz um esquema de perturbação de solução para evitar rotas de baixa qualidade. Tratando do problema clássico de programação de veículos onde um conjunto de veículos com capacidades conhecidas deve atender a um conjunto conhecido de pontos com demandas determinísticas ao menor custo. Resultados de experimentos computacionais são fornecidos tanto para os problemas simétricos quanto para os não-simétricos.

Além dos procedimentos heurísticos, as meta-heurísticas também se tornaram populares para encontrar soluções quase ótimas e até ótimas globais para problemas combinatórios, como o PRVC. Alguns exemplos são algoritmos de *Tabu Route* (GENDREAU; HERTZ; LAPORTE, 1994) que lidam com o problema de roteamento de veículos com restrições de capacidade e comprimento de rota. O algoritmo considera uma sequência de soluções adjacentes obtidas removendo repetidamente um vértice de sua rota atual e reinserindo-o em outra rota. Isso é feito por meio de um procedimento de inserção generalizada previamente desenvolvido. Durante o curso do algoritmo, soluções inviáveis são permitidas. Testes numéricos em um conjunto de problemas de benchmark indicam que a busca tabu supera as melhores heurísticas existentes, e geralmente produz as soluções mais conhecidas.

Já o *Boneroute* (TARANTILIS; KIRANOUDIS, 2002) usa um método meta-heurístico denominado: *backtracking adaptive threshold accept* (BATA). Sua arquitetura envolve uma estrutura integrada de sistema de informações geográficas (GIS) e um sistema de gerenciamento de banco de dados relacional (RDBMS) equipado com recursos de comunicação interativa entre ferramentas de software periféricas. O SDSS foi desenvolvido

para plataformas Windows 98, com foco na detalhada rede rodoviária de Atenas.

Entre as meta-heurísticas mais populares, encontram-se os Algoritmos de busca tabu (TS), como o proposto por (TAILLARD, 1993) que apresenta dois métodos de partição que aceleram métodos de busca iterativa aplicados a problemas de roteamento de veículos, incluindo um grande número de veículos. Usando uma implementação simples de busca tabu como um método de busca iterativo, todas as soluções mais conhecidas para problemas clássicos foram encontradas. O primeiro método de partição (baseado em uma partição em regiões polares) é apropriado para problemas euclidianos cujas cidades são regularmente distribuídas em torno de um depósito central. O segundo método de partição é adequado para qualquer problema e é baseado na arborescência construída a partir dos caminhos mais curtos de qualquer cidade até o depósito. Finalmente, soluções que se acredita serem ótimas são dadas para problemas gerados em uma grade.

Outro Algoritmo de TS é o proposto por (TOTH; VIGO, 2003) descrevendo uma nova variante. O método usa uma ferramenta efetiva de intensificação/diversificação que pode ser aplicada com sucesso a uma ampla classe de problemas de teoria de grafos e otimização combinatória. A busca tabu granular é baseada no uso de vizinhanças drasticamente restritas, não contendo os movimentos que envolvem apenas elementos que provavelmente não pertencem a boas soluções viáveis. Essas vizinhanças restritas são chamadas de granulares, e pode ser visto como uma implementação eficiente de estratégias de lista de candidatos propostas para algoritmos de busca tabu. Resultados de testes computacionais da abordagem proposta no bem conhecido problema de roteamento de veículos capacitado simétrico e com restrição de distância são discutidos, mostrando que a abordagem é capaz de determinar soluções muito boas em tempos de computação curtos.

Outros algoritmos que desempenham papel importante para este tipo de problema são os algoritmos genéticos (AG). Entre eles os propostos por (BERGER; BARKAOUI, 2003) e o (MESTER; BRÄYSY, 2007). O primeiro propõe um algoritmo genético híbrido para resolver o problema de roteamento de veículos capacitado. O esquema básico consiste em desenvolver simultaneamente duas populações de soluções para minimizar a distância total percorrida, usando operadores genéticos, combinando variações de conceitos-chave inspirados em técnicas de roteamento e estratégias de busca, usadas para uma variante temporal do problema para fornecer orientação de busca enquanto equilibra intensificação e diversificação . Os resultados de um experimento computacional sobre problemas comuns de *benchmark* relatam que a abordagem proposta é muito competitiva com os métodos mais conhecidos.

O segundo, apresenta uma adaptação da meta-heurística de estratégias de evolução guiada ativa para o problema de roteamento de veículos capacitado. A meta-heurística aplicada combina os pontos fortes das conhecidas meta-heurísticas de busca local guiada e estratégias de evolução em um procedimento iterativo de dois estágios. Os experimentos

computacionais foram realizados em um conjunto de 76 problemas-teste. Os resultados demonstram que o método sugerido é altamente competitivo, fornecendo as soluções mais conhecidas para 70 casos.

É importante ressaltar que, após a revisão bibliográfica para este trabalho, foi observado que apesar de vários trabalhos na literatura relatarem heurísticas/meta-heurísticas interessantes para a resolução do PRVC, poucos trabalhos abordam instâncias ou cenários baseados em dados geográficos reais. Em muitos casos, os autores posicionaram pontos aleatoriamente em um plano cartesiano e adotaram a distância euclidiana entre dois pontos.

2.3 Heurísticas e meta-heurísticas

Existem diversos problemas de otimização que necessitam de uma resolução que difere dos métodos matemáticos/exatos, tanto pela complexidade de resolução quanto pela necessidade de obter resultados em tempo viável. Dessa maneira, nas últimas décadas houve grande foco de pesquisas em métodos que possam resolver problemas de otimização de forma mais eficiente. Com isso, surgem os estudos de heurísticas e meta-heurísticas.

Heurística vem do grego *Heurikein* que significa descobrir e é uma técnica que se baseia em melhorar a eficiência em processos de busca de soluções. Uma heurística não é capaz de garantir a solução ótima, porém seus algoritmos são capazes de retornar soluções com qualidade em tempo adequado. Assim, as heurísticas exploram informações dependentes do problema para encontrar uma solução "suficientemente boa". As heurísticas podem ser classificadas em:

- Construtivas: processo iterativo que inicia com uma solução vazia e adiciona um novo elemento a cada iteração até a obtenção de uma solução.
- Decomposição: consiste em dividir o problema em subproblemas menores, de modo que a resolução de todos os subproblemas possam compor uma solução para o problema maior.
- Redução: identificam algumas características que presumidamente deverá possuir a solução ótima e simplifica o problema fixando-se o valor de tais variáveis.
- Modelo: modificam o modelo de tal forma que ele fique mais fácil de resolver.
- Busca: inicia em uma solução (podendo ser obtida a partir de outra heurística) e caminha sobre as soluções vizinhas.

"As meta-heurísticas são estruturas algorítmicas gerais adaptáveis a diversos problemas de otimização"(SUCUPIRA, 2004). Ou seja, as meta-heurísticas são um conjunto

de conceitos para definir métodos heurísticos que resolvem de forma genérica e adaptável problemas de otimização. Contudo, vale salientar que as meta-heurísticas não são universais, sendo assim não resolvem todos os problemas de otimização e sua aplicação requer conhecimentos específicos e fortes sobre o assunto onde será aplicada.

> Meta-heurísticas são aplicadas para encontrar respostas a problemas sobre os quais há poucas informações: não se sabe como é a aparência de uma solução ótima, há pouca informação heurística disponível e força-bruta é desconsiderada devido ao espaço de solução ser muito grande. Porém, dada uma solução candidata ao problema, esta pode ser testada e sua qualidade, averiguada (FRANCESQUINI,).

As meta-heurísticas alcançam o sucesso ao conseguir promover um balanceamento entre exploração e diversificação (LIMA, 2018). Ou seja, deve-se conseguir maximar o espaço de busca para obter diversidade nas soluções e também, buscar melhorar as soluções encontradas. Assim, as meta-heurísticas também podem ser classificadas, como segue abaixo (LEJBMAN, 2009):

- Meta-heurísticas de Métodos de Relaxação: modifica-se a modelagem do problema original, normalmente a função objetivo e/ou as restrições para criar um problema relaxado cuja solução é encontrada de forma eficaz.
- Meta-heurísticas para Processos Construtivos: Segue o mesmo princípio da heurística construtiva. A partir de uma solução inicial vazia, gradativamente é incrementada a solução parcial, de forma a não tornar-se inviável, até que o critério de parada seja atingido.
- Meta-heurísticas de Busca por Entornos: utiliza-se um espaço de busca formado por soluções (viáveis ou não) onde o algoritmo percorre esses espaços em busca de uma solução final, contudo, em cada iteração leva em consideração a solução anterior.
- Meta-heurísticas para Métodos Evolutivos: São algoritmos baseados a teoria evolutiva Darwiniana, onde para encontrar a melhor solução promovem a evolução da população através da reprodução de seus indivíduos.
- Meta-heurísticas Híbridas: são as meta-heurísticas que utilizam mais de uma das abordagens descritas acima ou apresentam abordagens completamente diferentes das vistas anteriormente.

2.4 Meta-heurística GRASP

Procedimentos de busca gulosos, aleatórios e adaptativos (do inglês, *Greedy Randomized Adaptive Search Procedure* - GRASP) é uma meta-heurística proposta por

(RESENDE; RIBEIRO, 2016) que aplica o método de busca local repetidamente a partir de soluções geradas por uma busca gulosa aleatória, usada para resolver problemas complexos e de grande porte.

A meta-heurística GRASP é considerada uma evolução para os processos construtivos, visto que esses processos visam construir uma solução gradativamente, utilizando um parâmetro ou indicador para escolher o melhor elemento a ser introduzido na solução.

O principal ponto da evolução do GRASP em relação aos outros processos é a aleatoriedade para escolher o elemento a ser adicionado na solução. A vantagem desse componente aleatório é atender ao caráter guloso da fase de construção da solução e também encontrar soluções de boas qualidades que se aproximem do ótimo (OLIVEIRA, 2011).

O GRASP apresenta uma abordagem em duas fases: construtiva e de busca local. A fase construtiva é onde aplica-se a busca gulosa para criar uma solução viável para o problema, que posteriormente será melhorada. A fase de busca local é onde ocorre o refinamento (ou melhoria) da solução obtida na fase de construção, com o uso da busca local, caso seja possível.

O Algoritmo 1 apresenta um pseudo-código do GRASP. Esse algoritmo pode ser dividido em quatro partes: na primeira é declarado o critério de parada assim como a leitura das instâncias; na segunda, é construído o primeiro conjunto de soluções com base nos dados iniciais, utilizando o algoritmo de busca gulosa; na terceira é realizado, se possível, o refinamento do resultado obtido na fase de construção, utilizando a busca local; na última fase ocorre o teste de melhor solução e de finalização do algoritmo, que geralmente é o número máximo de iterações.

Algorithm 1 GRASP

```
1: Entrada: Críterio de parada
2: Saída: Solução x*
3: while Criterio de parada nao satisfeito do
4:     x ← buscaGulosa()
5:     if x nao for viavel then
6:         x ← reparo(x)
7:     end if
8:     x ← buscaLocal(x)
9:     if f(x) < f(x*) then
10:        x* ← x
11:    end if
12: end while
13: return x*
```

As iterações do algoritmo são encontradas entre as linhas 3 e 12 e terminam quando um critério de parada é atingido. Na linha 4 ocorre a fase de construção da solução, enquanto a linha 6 é responsável por efetuar um procedimento de reparo quando a solução

construída não é viável. Na linha 8 a fase de busca local é iniciada e, por fim, se uma solução melhor foi encontrada, na linha 10 esta solução é guardada. A melhor solução é então retornada na linha 13.

O fator de aleatoriedade do GRASP é dado com a adição de um parâmetro α, que é passado para o algoritmo de busca gulosa na fase de construção. Essa adição é o ponto chave de uma busca gulosa aleatória.

2.4.1 Fase de construção

Na meta-heurística GRASP é na fase de construção onde são criadas diversas soluções viáveis que serão refinadas na fase de busca local. Comumente, em algoritmos gulosos, esta fase utiliza uma busca gulosa simples, em que não há diversidade entre as soluções, podendo o algoritmo ficar preso em "ótimos locais".

Contudo, existem versões em que nesta fase pode ser incluído um elemento de aleatoriedade que auxilia na diversidade de soluções. A qualidade das soluções, nos dois casos, é dada pela função gulosa que avalia e escolhe os elementos da solução.

A busca gulosa simples, consiste em ver o elemento que seja melhor para a solução, neste caso, não se leva em consideração o todo. O algoritmo 2 mostra um pseudo-código do que seria uma busca gulosa sem o fator de aleatoriedade.

Algorithm 2 Busca Gulosa Simples

1: Entrada: $E = \{conjunto\ discreto\ finito\}$
2: Saída: Solução S
3: $S \leftarrow 0$
4: **while** solução não construída **do**
5: Para todo c ∈ C computar o valor da função gulosa g(c);
6: Selecionar o melhor $c* \in$ g(c);
7: Adicionar c* à solução parcial: $S \leftarrow S \cup c*$;
8: **end while**
9: **return** S

Nas linhas 3 a 6 a solução é construída iterativamente. A cada iteração, na linha 5 será calculado o valor da função gulosa para cada um dos candidatos determinando seu grau de adaptação. Na linha 6 ocorre a escolha do melhor candidato. Na linha 7 o candidato é adicionado à solução e, por fim, é retornada a solução completa.

Ao adicionarmos um parâmetro $\alpha \in [0, 1]$ nessa fase de construção, além da diversidade de soluções, a qualidade também pode melhorar. Isso ocorre pois o elemento para ser incluído na solução, é escolhido aleatoriamente a partir de uma Lista de Candidatos Restrita (LCR), definida a partir do parâmetro α. Essa lista contém os melhores candidatos a serem incluídos na solução. O Algoritmo 3 mostra um pseudo-código do que seria uma busca gulosa com o α - o fator de aleatoriedade.

Algorithm 3 Busca Gulosa Aleatoria (α)

1: Entrada: $E = \{conjunto\ discreto\ finito\}$
2: Saída: Solução S
3: $S \leftarrow 0$
4: $C \leftarrow 0$
5: **while** s **do**olução não construída
6: Para todo c ∈ C computar o valor da função gulosa $g(c)$;
7: Construir a lista restrita de candidatos LRC com base no parâmetro α;
8: Selecionar aleatoriamente $c* \in LRC(C)$;
9: Adicionar c* a soluçao parcial: $S \leftarrow S \cup c*$;
10: Seja C o conjunto de elementos que podem ser adicionados a S;
11: **end while**
12: **return** S

O que difere a busca gulosa simples da aleatória, é o parâmetro α. No Algoritmo 3 é possível observar na linha 7 a utilização do α, seu intuito é construir a LRC, de maneira a selecionar os melhores elementos a serem inseridos na solução. A partir da construção dessa lista, é escolhido aleatoriamente um dos elementos que a ela pertencem e esse é incorporado à solução.

2.4.2 Fase de busca local

As soluções obtidas na fase de construção, geralmente não são ótimas. Desse modo, afim de melhorar as soluções obtidas, a segunda fase do GRASP se inicia, a fase de busca local.

Buscas locais, também referidas na literatura como buscas na vizinhança, são procedimentos utilizados para melhorar uma solução viável. Partindo de uma solução inicial, a cada iteração o algoritmo compara a solução atual com uma solução vizinha, no intuito de melhorar o valor da função objetivo. Uma das vantagens da busca local é poder encontrar soluções "boas" ou razoáveis em grandes conjuntos; além disso, ocupam pouca memória.

> A busca local pode ser vista como partindo de um vértice s ∈ S e examinado os nós adjacentes no grafo X em busca de uma solução de melhoria. (MATEUS; RESENDE; SILVA, 2010)

As estratégias para verificar a vizinhança são diversas, assim como o critério de parada. O sucesso dessa fase consiste em primeiro lugar, iniciar sobre soluções boas, ou seja, a fase de construção precisa gerar soluções de boa qualidade. O algoritmo 4 mostra um pseudo-código da busca local.

No Algoritmo 4, a variável x é inicializada com o valor da solução obtida na fase de construção, que é passada por parâmetro. Da linha 2 a 7 ocorre um *loop*, onde na linha 3 é realizada uma verificação com o intuito de descobrir se uma solução vizinha s melhora

Algorithm 4 Busca Local (*Solucao*)

```
1: x ←Solucao
2: while c doriterio Parada Nao Atingido
3:     if existe s ∈ N(x) tal que f(s) < f(x) then
4:         x ←s
5:     end if
6: end while
7: return x
```

o valor da função objetivo, caso isso ocorra, a variável x recebe a solução vizinha s. A melhor solução é retornada ao final.

3 Metodologia

Este trabalho tem por objetivo propor a utilização da meta-heurística GRASP para a resolução do PRVC. Para tanto, algumas definições precisam ser dadas afim de exemplificar como foi feita a especialização desta meta-heurística genérica para o problema em questão. Os dados de entrada e seu formato, os parâmetros do algoritmo, a definição do método de construção e de refinamento, assim como o método para encontrar a distância entre os pontos são fornecidos nas próximas seções.

3.1 Leitura da instância

Os dados utilizados para criar a instância de teste, foram cedidos pela empresa Loggi. Esses dados continham as informações de *ids*, longitude, latitude, capacidade do veículo e quantidade de entregas a serem realizadas naquele ponto e estavam em formato *JSON* (LOGGI, 2021). *JSON* é um formato de texto derivado da linguagem Javascript sendo representado por uma coleção de objetos compostos por pares de chave-valor (SILVA; JÚNIOR, 2018).

Para facilitar a leitura desses dados pelo algoritmo, realizamos a transformação do arquivo *JSON* para um novo formato: *.txt*. Além de alterar o formato do arquivo, utilizou-se uma nova estrutura, com o mesmo intuito de facilitar a leitura da instância. Dessa maneira, o arquivo de instância está estruturado da seguinte maneira: quantidade de pontos de entregas, seguido pela capacidade dos veículos, quantidade de pedidos e por fim, uma matriz que representa a distância de um ponto para os demais, como mostrado na Figura (5). Esse arquivo será o grafo de entrada para o algoritmo GRASP, pois nele conterá todas as informações da instância.

instancias_5.txt ×

src > matriz_distancias > arquivos > distancia > instancias_5.txt

```
1   5 <- nº de cidades
2   0
3   6
4   8
5   12
6   4
7   1000.0          0.3          1.1          1.5          9.0
8   0.3           1000.0         1.2          1.6          9.1
9   1.1             1.2        1000.0         1.7          9.2
10  1.5             1.6          1.7        1000.0         8.3
11  9.0             9.1          9.2          8.2        1000.0
```

Figura 5 – Exemplo de instância com cinco pontos de entrega. Fonte: Própria do autor.

Como o arquivo cedido pela empresa apresentava uma demasiada quantidade de pontos, foi usada apenas uma pequena amostra desses dados. Sendo, assim a capacidade do veículo presente nos dados, não condizia com a amostra realizada. Dessa forma, utilizamos um valor aleatório para representar a capacidade, de forma que não fosse retirada a complexidade do problema.

Além disso, a quantidade de veículos não foi informada, então, para termos a quantidade de veículo mais contundente, considerou-se como base o total de demandas, conforme a Equação (3.1):

$$\text{razão} = totalDemandas/(K \times Q) \tag{3.1}$$

Considerando K o número de veículos e Q a capacidade dos veículos, a *razão* deve ser o mais próximo possível de 1, tendo como consequência o número mínimo de veículos para se obter uma solução viável, e portanto, um problema mais difícil de resolver.

3.2 API Distance Matrix

Para obter a distância entre dois pontos, normalmente é usada a distância euclidiana, que é a distância entre dois pontos em linha reta (NETO, 2013). Contudo, para construir a matriz de distâncias entre os pontos neste problema, é necessário a distância real. Por este motivo, foi empregada a Distance Matrix API do Google (GOOGLE, 2021).

Esta API é incorporada diretamente no projeto e usa como parâmetros a longitude e latitude dos pontos que se deseja saber a distância. O algoritmo (5) mostra o pseudo-código da utilização dessa API neste estudo de caso.

Algorithm 5 ObterDistanciaReal

1: $gmaps \leftarrow googlemaps.Client(key =' xxx')$
2: $ObterDistanciaReal \leftarrow gmaps.distance\_matriz((latX, longX), (latY, longY))$
3: $distancia \leftarrow ObterDistanciaReal['distance']['text']$
4: $distancia \leftarrow (distancia.replace(", ", ".") * 1000)$
5: **return** *distancia*

Na linha 1 é onde o projeto se conecta com a API, na linha 2 é feito a chamada da função da API passando por parâmetro a longitude e latitude dos pontos X e Y. Esta API retorna diversos dados relativos a distância real entre dois pontos, contudo aqui utilizaremos apenas a distância real a ser percorrida, o que acontece na linha 3. A linha 4 mostra a padronização das distâncias para a mesma unidade de medida (quilômetros), com o intuito de facilitar a manipulação dos dados pelo algoritmo.

Vale ressaltar que a distância de um ponto para ele mesmo foi definida como 10.000,00 km, para que nos algoritmos de construção nunca fosse considerado o trajeto de um ponto para ele mesmo.

3.3 GRASP

Considerando o Problema do Roteamento de Veículos Capacitado e suas restrições, temos que a função objetivo consiste em minimizar o custo das rotas, nesse caso, custo equivale a distância a ser percorrida. Para isso, o Algoritmo 6 mostra um pseudo-código do GRASP que foi utilizado.

Algorithm 6 GRASP $(\alpha, MaxIterations)$

```
1: custo(melhorSolucao) ← ∞
2: for i = 0, 1, ..., MaxIterations do
3:     solucaoGulosa ← BuscaGulosa(α)
4:     solucao ← BuscaLocal(solucaoGulosa)
5:     if custo(solucao) < custo(melhorSolucao) then
6:         melhorSolucao ← solucao
7:     end if
8: end for
9: return bestSolution
```

O Algoritmo 6 recebe como parâmetros o $MaxIterations$ e o α, sendo $MaxIterations$ o número de iterações a serem executadas. Na linha 1, o valor da melhor solução é inicializado com um valor muito alto (∞), em seguida, das linhas 2 a 8, são executadas $MaxIterations$ iterações, no qual uma solução é construída de maneira gulosa e refinada com uma busca local. A cada iteração é feito um teste que verifica se a solução obtida é melhor que a melhor solução encontrada até o momento, e caso positivo, o valor da melhor solução é atualizado.

3.4 Busca Gulosa Aleatória

Para este trabalho, o PRVC precisa atender algumas restrições (mostradas na seção 2.1.1) e estas são adaptadas na fase de construção, ou seja, são necessárias para entender e construir o algoritmo de busca gulosa. Além disso, a busca gulosa utiliza a Lista de Candidatos Restrita, esta lista inclui todos os vértices que satisfazem as restrições e também, possuam um valor de custo incremental igual ou inferior ao *limite*, conforme a Equação (3.2):

$$limite = menorDistancia + \alpha * (maiorDistancia - menorDistancia) \tag{3.2}$$

O Algoritmo 7, mostra o pseudo-código da busca gulosa e é possível observar o papel da Lista de Candidatos Restrita.

Algorithm 7 BuscaGulosa $(\alpha, n\_veiculos, n\_vertices)$

1: $S \leftarrow \emptyset$ ▷ inicializando a solução
2: **for** $k = 0$ até $k < n\_veiculos$ **do**
3: $LC \leftarrow i \in V$: i é um vértice ainda não visitado
4: $LRC \leftarrow$ `criar_LRC`(α, LC, k) ▷ a partir da Equação (3.2)
5: $v \leftarrow$ `random`(LRC) ▷ escolhendo um vértice para ser visitado por k
6: $S[k] \leftarrow S[k] \cup v$
7: **end forreturn** S

No Algoritmo de Busca Gulosa, na linha 1 é declarado a variável *solucao* como vazia, ela representa a solução que será construída. Na linha 2 é iniciado um *loop* de zero até o número de veículos e na linha 3 é iterado sobre o número de vértices. Na linha 4 é criada a LCR, e nesta é utilizado o parâmetro α. Com a lista de candidatos finalizada, é escolhido um elemento aleatório que fará parte da solução, como mostra a linha 5. Ao final das iterações, é retornada a solução construída que será refinada na busca local.

No Algoritmo 7, iteramos tanto sobre os veículos quanto sobre os vértices, de modo a obter soluções considerando a quantidade de veículos e suas capacidades. Para cada veículo, inicialmente são considerados todos os vértices ainda não visitados (conjunto LC). Em seguida, é cria uma lista de candidatos restrita (LCR), que é composta pelos vértices que obtêm custo incremental menor ou igual ao *limite* e que a demanda não ultrapassa a capacidade atual do veículo. Após a lista finalizada, aleatoriamente é escolhido um dos vértices de LRC que será incorporado à solução gulosa que está sendo montada. Ao final do laço que percorre todos os veículos, a solução construída é retornada.

3.5 Busca Local

Com a fase de construção finalizada, inicia a fase adaptativa ou de busca local. Agora é a vez de realizar um refinamento da solução obtida na fase anterior com o intuito de melhorar o custo, ou seja, minimizá-lo caso possível. Pretende-se reduzir o custo das rotas, caso seja possível.

Esta busca local apresenta o seguinte método para alcançar o objetivo geral: troca de posição dos vértices da rota já pré-estabelecida, para verificar se a alteração da ordem dos vértices reduz o custo total. O Algoritmo 8 é um pseudo-código da busca local realizada nesse estudo de caso.

Na linha 1 é declarada á *solucaoRefinada* que será usada para comparar se as trocas realizadas estão realmente diminuindo o custo. Na linha 2 ocorre um *loop* de cada rota da solução, na linha 3 é pego o último elemento da rota a ser verificada. Da linha 4 a 12 temos uma nova iteração, dessa vez apenas dos elementos presente na rota atual.

Algorithm 8 BuscaLocal (S)

```
1: S_refinada ← S
2: for k = 0 até k < n_veiculos do
3:     for v = 0 até v < tamanho(S[k]) − 1 do
4:         custo_k_antes ← custo(S[k])
5:         troca(S[k][v], S[k][v + 1])                ▷ trocando a ordem de visita na rota k
6:         custo_k_depois = custo(S[k])
7:         if custo_k_depois ≥ custo_k_antes then
8:             troca(S[k][v + 1], S[k][v])                              ▷ desfazendo a troca
9:         else
10:            S_refinada ← S
11:        end if
12:    end for
13: end for
14: return S_refinada
```

Na linha 5 é calculado o custo total da rota e utilizando de uma variável auxiliar (linha 6) é realizada a troca da posição dos vértices dentro de uma rota e refeito o cálculo do custo total após a troca (linhas 7 e 8). Caso, essa troca tenha minimizado o custo, essa troca se torna permanente (linhas 9 a 11). Ao final, após testar todas as trocas, dentro de cada rota, é retornada a solução refinada.

Para este trabalho a busca local ocorreu da seguinte forma. Para cada veículo e para cada vértice pertencente à rota desse veículo, é verificado se uma troca de ordem de visita entre dois vértices melhora o custo da solução. Caso melhore, a solução refinada da busca local é atualizada. Por fim, a solução $S\_refinada$ é retornada.

4 Resultados computacionais

Para a realização dos experimentos deste trabalho, a linguagem escolhida foi o python 3.8 e o ambiente foi um computador pessoal Intel® Core™ i5 CPU@1,60GHz com 8GB RAM, utilizando o sistema operacional Linux Ubuntu 18.04.5. A seguir, são apresentados experimentos computacionais sobre a calibração de parâmetros da heurística GRASP proposta, e em seguida é realizada uma comparação com um método de busca gulosa.

4.1 Calibração de parâmetros

Para se obter uma melhor performance do método proposto, proporção de custo por tempo computacional, foram realizados experimentos com os parâmetros α e $MaxIterations$ do GRASP, respectivamente.

Para a calibração desses parâmetros, foram utilizadas instâncias com 75 e 125 vértices com a quantidade de veículos definida conforme a Equação (3.1) e suas respectivas capacidades, baseadas em dados reais dos estados do Rio de Janeiro (RJ), Distrito Federal (DF) e Pará (PA). As Tabelas 1, 2, 3 mostram a calibração em sua primeira fase, ou seja, os testes com o parâmetro α. As três primeiras colunas representam os dados referentes às instâncias, enquanto as três últimas colunas referem-se ao algoritmo GRASP e seus resultados.

Tabela 1 – Tabela de calibração de parâmetros com $\alpha = 0{,}70$ e 10 iterações

			GRASP		
Qtd. Pontos	Q	**UF**	**Custo médio**	**Menor custo**	**Tempo médio (s)**
75	37	RJ	142,51	135,90	2,73
	37	DF	693,35	609,90	2,79
	37	PA	362,94	357,10	2,86
125	62	RJ	237,77	231,50	6,93
	62	DF	1.218,66	1.088,50	7,20
	62	PA	660,00	630,00	8,63

Com base nas Tabelas 1, 2, 3, observa-se que o tempo médio gasto é similar em todos os casos. Sendo assim, conclui-se que a variação do parâmetro α não influencia no tempo de execução. Contudo, em relação ao custo - nesse caso a distância total a ser percorrida pelas rotas - observa-se melhores resultados com o parâmetro α igual a 0,70. A Tabela 4 e a Tabela 5 apresentam uma comparação mais direta entre os valores de custo obtidos.

Tabela 2 – Tabela de calibração de parâmetros com $\alpha = 0{,}80$ e 10 iterações

			GRASP		
Qtd. Pontos	Q	**UF**	**Custo médio**	**Menor custo**	**Tempo médio (s)**
75	37	RJ	147,57	139,39	2,59
	37	DF	877,06	787,90	2,88
	37	PA	365,15	357,60	2,90
125	62	RJ	244,07	230,10	6,90
	62	DF	1.382,40	1.247,20	7,12
	62	PA	669,39	638,60	7,07

Tabela 3 – Tabela de calibração de parâmetros com $\alpha = 0{,}90$ e 10 iterações

			GRASP		
Qtd. Pontos	Q	**UF**	**Custo médio**	**Menor custo**	**Tempo médio (s)**
75	37	RJ	150,37	137,70	2,77
	37	DF	900,55	845,00	2,75
	37	PA	387,93	364,00	3,11
125	62	RJ	262,59	251,09	7,38
	62	DF	1.402,15	1.277,79	7,11
	62	PA	684,58	648,19	7,29

Tabela 4 – Tabela de comparação dos melhores casos para a calibração de α

Qtd. Pontos	**UF**	α**= 0,70**	α**= 0,80**	α **= 0,90**	**Melhor caso**
75	RJ	135,90	139,39	137,70	α **= 0,70**
	DF	609,90	787,90	845,00	α **= 0,70**
	PA	357,10	357,60	364,00	α **= 0,70**
150	RJ	231,50	230,10	251,09	α **= 0,80**
	DF	1.088,50	1,247,20	1.277,79	α **= 0,70**
	PA	630,00	638,60	648,69	α **= 0,70**

A partir da Tabela 4 e Tabela 5, verifica-se que a configuração do GRASP com $\alpha = 0,70$ é superior as demais, chegando a ser melhor em aproximadamente 84% das instâncias. Dessa forma, foi definido $\alpha = 0,70$ como sendo a configuração de α padrão para a heurística proposta. A próxima fase de calibração é sobre o parâmetro referente a quantidade máxima de iterações. Para isso, com o α fixo em 0,70, o número de iterações utilizado nos primeiros testes (10) foi duplicado e triplicado, conforme apresentado nas Tabelas 6 e 7.

Como mencionado anteriormente, o tempo de execução da heurística proposta é diretamente proporcional a sua quantidade de iterações, o que implica em um aumento da probabilidade de se obter uma solução de melhor qualidade. Portanto, o ponto-chave dos experimentos com o parâmetro $MaxIterations$ é verificar o quanto podemos incrementar o número de iterações sem que o método se torne computacionalmente inviável ou que o

Tabela 5 – Tabela de comparação das médias para a calibração de α

Qtd. Pontos	**UF**	α**= 0,70**	α**= 0,80**	α **= 0,90**	**Melhor caso**
75	RJ	142,51	147,57	150,37	α **= 0,70**
	DF	693,35	877,06	900,55	α **= 0,80**
	PA	462,94	365,15	387,93	α **= 0,80**
125	RJ	237,77	244,07	262,59	α **= 0,70**
	DF	1,218,66	1.382,40	1.402,15	α **= 0,70**
	PA	660,00	669,39	648,58	α **= 0,70**

Tabela 6 – Tabela de calibração de parâmetros com $\alpha = 0{,}70$ e 20 iterações

			GRASP		
Qtd. Pontos	Q	**UF**	**Custo médio**	**Menor custo**	**Tempo médio (s)**
75	37	RJ	140,21	134,80	3,10
	37	DF	715,43	681,19	3,00
	37	PA	360,96	354,70	3,13
125	62	253,96	234,35	226,69	7,33
	62	RJ	1.118,44	1.015,50	7,12
	62	PA	651,05	628,40	7,08

Tabela 7 – Tabela de calibração de parâmetros com $\alpha = 0{,}70$ e 30 iterações

			GRASP		
Qtd. Pontos	Q	**UF**	**Custo médio**	**Menor custo**	**Tempo médio (s)**
75	37	RJ	137,69	132,80	5,07
	37	DF	718,52	132,80	5,30
	37	PA	356,57	345,59	5,43
125	62	RJ	234,26	224,60	14,13
	62	DF	1.202,24	892,09	13,66
	62	PA	637,85	602,50	13,33

cliente aguarde demasiado tempo para uma resposta. A Tabela 8 apresenta uma comparação entre os melhores resultados obtidos com a variação da quantidade de iterações.

Para este primeiro ponto, a Tabela 9 mostra uma comparação entre as médias obtidas com a variação da quantidade de iterações. Essa tabela deixa claro que, na maioria dos casos, o aumento da quantidade de iterações melhorou o custo das rotas, equivalente a 75% dos casos.

A partir da Tabela 8, verificamos que de fato, o aumento da quantidade de iterações melhorou o custo das rotas, em 75% dos casos. Por outro lado, em relação ao tempo para encontrar essas soluções, com 30 iterações, praticamente dobra em todos os casos, resultando em um tempo computacional considerado limite para o tipo de solução buscada.

Tabela 8 – Tabela de comparação dos melhores casos para a calibração do número de iterações

Qtd. Pontos	**UF**	**Iteração = 10**	**Iteração = 20**	**Iteração = 30**	**Melhor caso**
75	RJ	135,90	134,80	132,80	**I = 30**
	DF	609,90	681,19	622,09	**I = 10**
	PA	357,10	354,70	345,59	**I = 30**
125	RJ	231,50	226,69	224,60	**I = 30**
	DF	1.088,50	1.015,50	892,09	**I = 30**
	PA	630,00	628,40	602,50	**I = 30**

Tabela 9 – Tabela de comparação das médias para a calibração do número de iterações

Qtd. Pontos	**UF**	**Iteração = 10**	**Iteração = 20**	**Iteração = 30**	**Melhor caso**
75	RJ	142,51	140,21	137,69	**I = 30**
	DF	693,35	715,43	718,52	**I =20**
	PA	362,94	360,96	356,57	**I = 30**
125	RJ	237,77	234,35	234,26	**I = 30**
	DF	1.218,66	1.118,44	1.202,24	**I = 20**
	PA	660,00	651,05	637,85	**I = 30**

Dessa forma, a calibração dos parâmetros é encerrada com o α igual a 0,70 e 30 iterações no máximo.

4.2 GRASP vs Busca Gulosa

Com os parâmetros do algoritmo GRASP definidos, foram realizados também experimentos com um algoritmo de busca gulosa "simples". Neste contexto, por "simples" definimos uma heurística puramente gulosa e sem nenhum mecanismo para escapar de ótimos locais. Com esse experimento, buscamos verificar o impacto da abordagem proposta em relação a uma abordagem gulosa que geralmente é mais utilizada em situações reais na indústria. Os resultados desse experimento encontram-se na Tabela 10.

É perceptível que os resultados obtidos com o GRASP são bem mais otimizados que os resultados da Busca Gulosa, em um tempo computacional viável nos dois casos. Em 100% das instâncias testadas o GRASP mostrou resultados superiores ao da Busca Gulosa, em termos de qualidade de solução, apresentando reduções que chegam a 26%. A Tabela 10 apresenta um resumo das reduções obtidas, em sua última coluna.

É observado que as reduções são de no mínimo 8% e podem chegar até a aproximadamente 26%. No contexto de um cenário de simulação, que foi utilizado neste trabalho, uma redução de 26% é considerável. Já para um cenário real, que possui milhares de

Tabela 10 – Resultados de calibragem do parâmetro α (GRASP) X resultados da Busca Gulosa

			GRASP			**Busca Gulosa**		
Qtd. Pontos	Q	**UF**	**Custo médio**	**Menor custo**	**Tempo médio (s)**	**Menor custo**	**Tempo**	**Redução (%)**
50	25	RJ	92,80	91,40	2,00	123,80	1,56	26, 17
	25	DF	388,90	387,40	2,50	464,10	1,08	16,53
	25	PA	317,70	315,90	2,30	400,30	1,13	21,08
75	37	RJ	142,51	135,90	2,73	178,40	1,78	23,82
	37	DF	693,35	609,90	2,79	708,10	1,98	13,87
	37	PA	362,94	357,10	2,86	423,50	1,34	15,68
100	50	RJ	186,40	184,30	5,59	229,70	2,12	19,76
	50	DF	890,30	878,60	5,50	1.144,20	2,33	23,25
	50	PA	536,70	534,00	5,56	629,75	2,89	15,20
125	62	RJ	237,77	231,50	6,93	275,56	3,45	15,99
	62	DF	1.218,66	1.088,50	7,20	1.244,00	3,12	12,54
	62	PA	660,00	630,00	8,63	700,10	3,76	10,01
150	75	RJ	400,00	398,80	8,34	524,40	3,05	23,95
	75	DF	1.737,20	1.735,40	8,72	1.920,30	3,48	9,64
	75	PA	860,50	858,40	8,68	1.050,10	3,20	18,29
175	87	RJ	1.057,88	989,19	17,10	1.080,56	4,76	8,43
	87	DF	1.377,06	1.289,90	18,03	1.400,35	5,76	8,93
	87	PA	1.062,05	1.039,79	16,85	1.240,98	5,12	16,21
20	100	RJ	700,50	699,90	22,34	854,1	5,34	18,15
	100	DF	1.071,00	1.069,10	25,58	1.351,4	5,98	20,87
	100	PA	1.152,70	1.150,90	23,91	1.503,4	5,21	23,49

pontos e demandas, essa redução tende a ser ainda mais significativa, oferecendo uma minimização interessante dos custos operacionais de logística.

5 Conclusão

Neste trabalho foi aplicada a meta-heurística GRASP para a resolução do Problema de Roteamento de Veículos Capacitado. O PRVC é um problema de Otimização Combinatória do tipo NP-Difícil e resolvê-lo requer grande esforço, devido à sua complexidade.

Para a validação da proposta de solução, foram geradas instâncias a partir de uma base de dados real fornecida pela empresa Loggi, contendo as informações: quantidade de pontos, demandas, e matriz de distâncias. Informações sobre quantidade de veículos e suas capacidades foram obtidas por meio de fórmulas matemáticas, que garantem uma dificuldade aceitável de resolução do problema.

O método proposto neste trabalho mostrou-se como viável para a resolução do problema do roteamento de veículos capacitado. Obteve-se resultados satisfatórios e conseguiu-se atender todos os critérios elencados tanto pelo problema, quanto pela própria técnica e os recursos computacionais.

Além disso, este trabalho apresentou as vantagens do algoritmo GRASP de maneira comparativa com uma Busca Gulosa "simples". Desse modo, acredita-se que a solução apresentada neste trabalho pode auxiliar o setor logístico de empresas, a fim de se ter um melhor controle sobre a operação de transporte com base em dados reais.

É fundamental a realização de estudos complementares sobre a técnica e o problema. Além disso, é de suma importância que essas técnicas sejam cada vez mais incorporadas no âmbito empresarial, de modo a testá-las no contexto de uso diário, para que assim possam ser cada vez mais aperfeiçoadas.

Como trabalhos futuros, dado que o GRASP é um método multi-partida e "sem memória", pois informações de uma iteração não transmitidas a iterações futuras, pode-se combinar o GRASP proposto com mineração de dados para que se possa re-utilizar informações obtidas em iterações passadas. Neste trabalho foi explorado apenas uma vizinhança na fase de busca local, então um caminho natural para investigações futuras seria elaborar um maior número de vizinhanças e combiná-las em um algoritmo VND (*Variable Neighborhood Descent*).

A realização deste trabalho foi desafiadora. A base de conhecimentos em Pesquisa Operacional é um ponto forte e necessário, que ao longo do trabalho foi sendo buscada. O estudo foi modularizado, afim de tornar as partes menores e depois obtermos uma solução completa. O auxílio do orientador Francisco Glaubos foi de suma importância em todo o percusso deste trabalho.

Referências

AKHTAR, M. et al. Backtracking search algorithm in cvrp models for efficient solid waste collection and route optimization. *Waste Management*, Elsevier, v. 61, p. 117–128, 2017. Citado na página 1.

ARAÚJO, F. de; LIMA, A. A.; LIMA, M. d. A. C. Otimização de rota e redução dos custos logísticos: estudo de caso em uma empresa de contabilidade. *Brazilian Journal of Development*, v. 4, n. 1, p. 136–144, 2018. Citado na página 2.

ASSIS, L. P. de. Algoritmos para o problema de roteamento de veículos com coleta e entrega simultâneas. Universidade Federal de Minas Gerais, 2007. Citado na página 5.

BERGER, J.; BARKAOUI, M. A hybrid genetic algorithm for the capacitated vehicle routing problem. In: SPRINGER. *Genetic and evolutionary computation conference*. [S.l.], 2003. p. 646–656. Citado na página 8.

BITTENCOURT, G. C. de et al. Problema de roteamento de veículos capacitados (prvc): solução manual x busca dispersa. In: *Congresso Latino-Iberoamericano de Investigatión Operativa–CLAIO. Simpósio Brasileiro de pesquisa Operacional–SBPO: Rio de Janeiro*. [S.l.: s.n.], 2012. Citado na página 5.

CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, Informs, v. 12, n. 4, p. 568–581, 1964. Citado na página 6.

FRANCESQUINI, E. de C. Mac 5758-introdução ao escalonamento e aplicações escalonamento através de perfilamento em sistemas multi-core. Citado na página 10.

GENDREAU, M.; HERTZ, A.; LAPORTE, G. A tabu search heuristic for the vehicle routing problem. *Management science*, INFORMS, v. 40, n. 10, p. 1276–1290, 1994. Citado na página 7.

GOMES, J. A. C. et al. Aplicação de ferramenta computacional na otimização e mitigação de custos na roteirização da logística de transporte de cargas. *Brazilian Journal of Development*, v. 5, n. 7, p. 7703–7716, 2019. Citado na página 2.

GOOGLE. *Distance Matrix API*. 2021. <https://console.cloud.google.com/marketplace/product/google/distance-matrix-backend.googleapis.com?q=search&referrer=search>. Acesso 07 de outubro de 2021. Citado na página 16.

GUIMARÃES, T. A.; PEREIRA, L. G.; MEDEIROS, W. J. N. Simulação de monte carlo e métodos heurísticos paralelizados para a resolução do problema do roteamento de veículos capacitados: uma comparação entre abordagens. *SIMPÓSIO DE PESQUISA OPERACIONAL E LOGÍSTICA DA MARINHA–SPOLM*, v. 15, 2012. Citado na página 2.

HASLE, G.; LIE, K.-A.; QUAK, E. *Geometric modelling, numerical simulation, and optimization*. [S.l.]: Springer, 2007. Citado na página 2.

HOLMES, R.; PARKER, R. A vehicle scheduling procedure based upon savings and a solution perturbation scheme. *Journal of the Operational Research Society*, Springer, v. 27, n. 1, p. 83–92, 1976. Citado na página 7.

LEJBMAN, D. A. G. vel. Um estudo abrangente sobre metaheurística, incluindo um histórico. 2009. Citado na página 10.

LIMA, I. L. C. Meta-heurística grasp aplicada ao problema de localização de contadores de tráfego. Universidade Federal do Maranhão, 2018. Citado na página 10.

LOGGI. *Github LOGGI Instances*. 2021. <https://github.com/loggi/loggibud/tree/master/tests/test_instances>. Acesso 10 de outubro de 2021. Citado na página 15.

MATEUS, G. R.; RESENDE, M. G.; SILVA, R. M. Grasp: Procedimentos de busca gulosos, aleatórios e adaptativos. *2a Escola Luso-Brasileira de Computação Evolutiva*, 2010. Citado na página 13.

MENEGATTI, M. S. et al. Decisão de compras pela internet: uma análise a partir do tempo de utilização de mídias sociais e da interatividade com a marca. *Revista Brasileira de Marketing*, Universidade Nove de Julho, v. 16, n. 1, p. 41–54, 2017. Citado na página 1.

MESTER, D.; BRÄYSY, O. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & operations research*, Elsevier, v. 34, n. 10, p. 2964–2975, 2007. Citado na página 8.

MOLE, R.; JAMESON, S. A sequential route-building algorithm employing a generalised savings criterion. *Journal of the Operational Research Society*, Taylor & Francis, v. 27, n. 2, p. 503–511, 1976. Citado na página 7.

NETO, I. F. Um novo conceito de distância: a distância do táxi e aplicações. Universidade Estadual Paulista (UNESP), 2013. Citado na página 16.

OLIVEIRA, M. B. C. d. Reconfiguração de alimentadores em sistemas de distribuição usando a metaheurística grasp. Universidade Estadual Paulista (UNESP), 2011. Citado na página 11.

RESENDE, M. G.; RIBEIRO, C. C. *Optimization by GRASP*. [S.l.]: Springer, 2016. Citado na página 11.

RODRIGUE, J.-P.; COMTOIS, C.; SLACK, B. *The geography of transport systems*. [S.l.]: Routledge, 2016. Citado na página 2.

SANTOS, F. V. d. A utilização da pesquisa operacional como ferramenta para redução de custos na logística de distribuição: problema de roteamento de veículos capacitados (prvc). 2014. Citado na página 1.

SANTOS, J.; SANTOS, A.; BERTO, A. R. Logística: Evolução e perspectiva. *Revista de Ciências Empresariais*, v. 2, n. 4, p. 1–14, 2009. Citado na página 1.

SILVA, P. C. d.; JÚNIOR, J. B. d. S. Análise da representação semântica de modelos de dados do formato json. *Revista de Sistemas e Computação-RSC*, v. 8, n. 1, 2018. Citado na página 15.

SOUZA, V. A. A. de. Algoritmos para o problema de roteamento de veículos capacitado com restrições de carregamento bidimensional. Universidade Federal de Minas Gerais, 2013. Citado na página 5.

SUCUPIRA, I. R. Métodos heurísticos genéricos: metaheurísticas e hiper-heurísticas. *USP: São Paulo*, v. 32, 2004. Citado na página 9.

TAILLARD, É. Parallel iterative search methods for vehicle routing problems. *Networks*, Wiley Online Library, v. 23, n. 8, p. 661–673, 1993. Citado na página 8.

TARANTILIS, C. D.; KIRANOUDIS, C. T. Using a spatial decision support system for solving the vehicle routing problem. *Information & Management*, Elsevier, v. 39, n. 5, p. 359–375, 2002. Citado na página 7.

TOTH, P.; VIGO, D. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, INFORMS, v. 15, n. 4, p. 333–346, 2003. Citado na página 8.