UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE EDUCAÇÃO SUPERIOR DO ALTO VALE DO ITAJAÍ – CEAVI
CURSO DE ENGENHARIA DE SOFTWARE - ESO

RELATÓRIO DA ANÁLISE DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS
VEHICLE ROUTING PROBLEM (VRP)

JOÃO EDUARDO KRIEGER E MAX NATANAEL STARKE

ORIENTADOR
MARCELO DE SOUZA

IBIRAMA
JULHO/2023

1. PROBLEMA

1.1 DESCRIÇÃO DETALHADA DO PROBLEMA

O problema de *vehicle routing*, ou roteamento de veículos, é um desafio clássico de otimização logística. Ele envolve o planejamento eficiente de rotas para um conjunto de veículos que devem atender a um conjunto de clientes, cada um com suas próprias demandas, levando em consideração várias restrições, como capacidade dos veículos, janelas de tempo de atendimento e custos de deslocamento.

Nesse sentido, o objetivo principal do *vehicle routing* é minimizar os custos totais, que geralmente estão relacionados à distância percorrida pelos veículos ou ao tempo necessário para completar todas as entregas. Isso implica em encontrar uma alocação de clientes para veículos e sequências de visitas que otimizem a utilização dos recursos disponíveis e satisfaçam todas as restrições impostas.

No contexto aplicado de um grafo completo, todos os clientes estão conectados diretamente uns aos outros, ou seja, é possível ir de qualquer cliente a qualquer outro cliente diretamente, sem precisar passar por intermediários. Isso significa que há uma aresta entre cada par de clientes no grafo. Essa representação é vantajosa porque permite considerar todas as rotas possíveis entre os clientes. Além disso, todos os clientes possuem demanda, o que significa que cada cliente precisa receber uma certa quantidade de produtos. Essas demandas podem ser quantidades físicas, como o número de itens a serem entregues.

A consideração das demandas dos clientes é importante para garantir que a capacidade dos veículos seja respeitada. Cada veículo possui uma capacidade máxima de carga, e as demandas dos clientes devem ser acomodadas dentro dessa capacidade durante as visitas.

Em resumo, o problema de vehicle routing em um grafo completo, onde todos os clientes possuem demanda, envolve encontrar as rotas mais eficientes para um conjunto de veículos atenderem a todos os clientes, minimizando a distância total percorrida pelos veículos e otimizando a utilização dos recursos disponíveis, considerando as capacidades dos veículos e as demandas dos clientes.

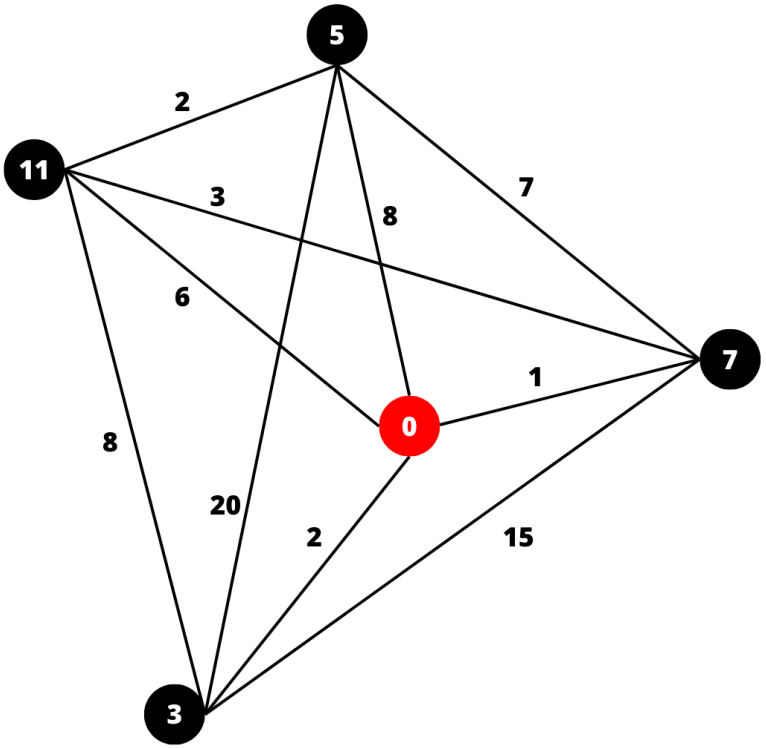
1.2 FORMULAÇÃO MATEMÁTICA

Ao considerar um grafo completo, assumimos que todos os clientes estão conectados diretamente uns aos outros, ou seja, é possível estabelecer

rotas diretas de um cliente para qualquer outro cliente, sem precisar passar por intermediários. Além disso, levaremos em conta as demandas dos clientes, que representam a quantidade de carga que cada cliente precisa receber.

Nesse sentido, apresentaremos uma análise visual do grafo do problema, demonstrando a distribuição dos clientes e as possíveis rotas de atendimento. Essa visualização permitirá uma compreensão mais intuitiva do problema e auxiliará na identificação de padrões e oportunidades de otimização.

Figura 1 - Análise visual do grafo VRP



Fonte: De autoria própria

Neste caso, consideramos que cada cliente é representado por um ponto (x, y) no plano cartesiano, o que nos permite visualizar claramente a localização de cada cliente. Portanto, a formulação matemática do problema, que leva em consideração a demanda dos clientes e a capacidade de carga dos veículos, pode ser expressa da seguinte forma:

Conjuntos e parâmetros introdutórios do problema
❖ **Conjuntos:** Conjunto de veículos **V** = {1,2,...,m}, onde m é o número total de veículos disponíveis.

Conjunto de clientes **C** = {1,2,..,n}, onde n é o número total de clientes.

❖ **Parâmetros:**

Capacidade máxima de carga do veículo k: **Qk**, para todo k∈V.

Demanda do cliente i: **Di**, para todo i∈**C**.

Matriz de coordenadas do cliente i: (xi, yi), para todo i∈**C**.

Variáveis de decisão

- **X_{ij}:** Variável binária que indica se o veículo k percorre a rota do cliente i ao cliente j. Assume o valor 1 se o veículo seguir essa rota e 0 caso contrário.
- **C_{ij}:** Variável contínua que representa o representa o custo de ir do nó i ao nó j.

Formulação matemática

❖ **Minimizar:** Encontrar a solução que resulte na menor distância total percorrida pelos veículos para atender a todas as demandas dos clientes.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

Restrições do problema

$$\sum_{i \in V} x_{ij} = 1 \quad \forall j \in V \setminus \{0\}$$

$$\sum_{j \in V} x_{ij} = 1 \quad \forall i \in V \setminus \{0\}$$

- Estabelecem que exatamente um arco entra e exatamente um sai de cada vértice associado a um cliente.

$$\sum_{i \in V \setminus \{0\}} x_{i0} = K$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = K$$

- Estabelecem que o número de veículos que saem do depósito é o mesmo que o número que entra.

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \ \ \forall S \subseteq V \setminus \{0\}, S \neq \emptyset$$

- Restrições de corte de capacidade, que impõem que as rotas sejam conectadas e que a demanda em cada rota não exceda a capacidade do veículo.

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V$$

- Restrições de integralidade

1.3 CARACTERÍSTICAS

O Problema de Roteamento de Veículos (VRP) é uma área de estudo amplamente reconhecida na otimização combinatória, este problema aborda o desafio de planejar rotas eficientes para a locomoção de veículos que precisam atender a um conjunto de locais de entrega, respeitando restrições operacionais. O VRP possui uma diversificada gama de aplicações em diversas indústrias, como logística, transporte público, distribuição de mercadorias, entre outras.

Em geral, o VRP é caracterizado pela operação eficiente de uma frota de veículos para realizar a entrega de produtos ou serviços aos clientes, levando em consideração suas demandas individuais. Além disso, o VRP pode envolver uma variedade de restrições operacionais específicas, adaptadas às necessidades de cada caso de uso.

Uma característica muito comum em aplicações do VRP é a inclusão de restrições temporais, como janelas de tempo para a conclusão das rotas de entrega. Essas janelas determinam um intervalo específico em que cada cliente deve ser atendido, levando em conta a disponibilidade e preferências dos mesmos. Essas restrições temporais adicionam complexidade ao problema, pois é necessário otimizar as rotas para atender a todas as demandas dos clientes dentro dos prazos estabelecidos.

Por fim, é importante ressaltar que cada instância do VRP pode ter suas próprias particularidades e restrições adicionais, como a capacidade dos veículos, limitações de distância ou custos operacionais. Portanto, a formulação e solução do VRP requerem uma análise cuidadosa de todos os fatores para encontrar uma solução eficiente.

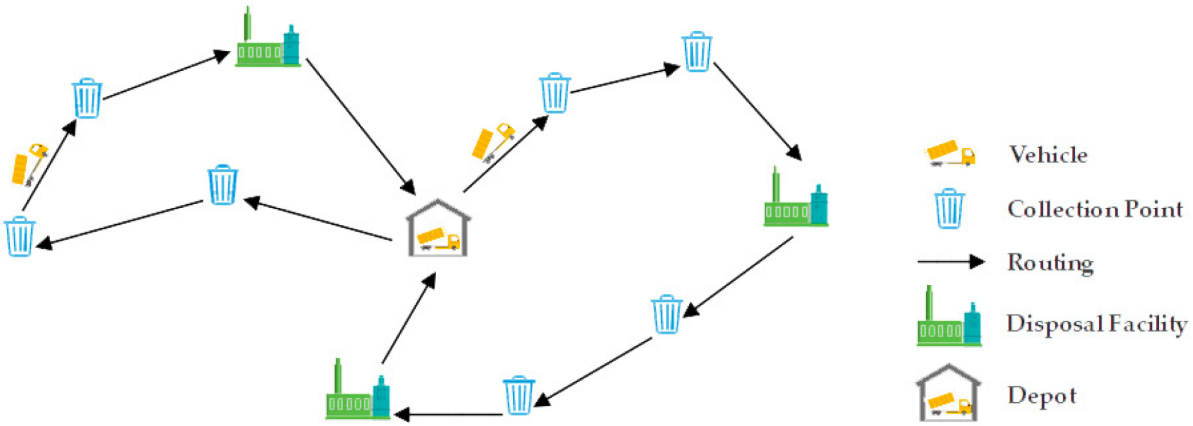
1.4 APLICAÇÕES

Os problemas de roteamento são extremamente relevantes na vida real, surgindo em diversas situações em que é necessário transportar itens ou pessoas de um local para outro de maneira eficiente e de custo benefício. Por exemplo, empresas de coleta de lixo enfrentam o desafio de planejar

rotas eficientes para coletar o lixo nas áreas urbanas, enquanto as empresas de transporte público precisam otimizar o tempo e as rotas dos ônibus, além de gerenciar os motoristas.

No caso das empresas de coleta de lixo, é fundamental criar rotas eficientes para garantir que todos os locais de coleta sejam atendidos no menor tempo possível. Isso requer a consideração de diversos fatores, como a localização dos pontos de coleta, a capacidade dos veículos e as restrições de tráfego.

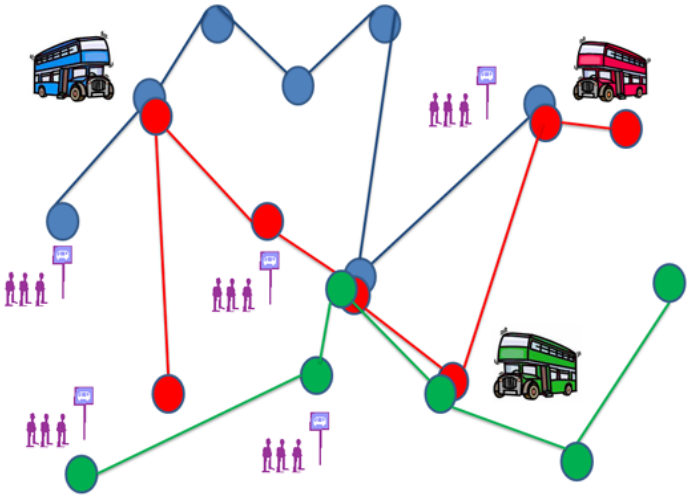
Figura 2 - Representação do Roteamento de Veículos para Coleta de Resíduos.



Fonte: MDI Collection Management, 2020.

Já as empresas relacionadas ao transporte público enfrentam o desafio de planejar horários e rotas que atendam às necessidades dos passageiros, levando em conta a demanda em diferentes horários e locais. Além disso, é importante considerar fatores como a distância percorrida, a eficiência do sistema de transporte e a capacidade de passageiros de cada veículo.

Figura 3 - Representação do Roteamento de Veículos para Transporte Público.



Fonte: Christine Mumford, 2009

Dessa forma, a resolução eficiente dos problemas de roteamento desempenha um papel fundamental na melhoria da logística e na prestação de serviços de transporte eficientes, seja na coleta e entrega ou na área do transporte.

2. INSTÂNCIAS

Durante a elaboração do estudo e da pesquisa, foram utilizadas instâncias fornecidas pela biblioteca pública da universidade PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro).

Essas instâncias são conjuntos de dados gerados que representam cenários diferentes do problema, como diferentes números de clientes, capacidades de veículos, distâncias e restrições. Dessa forma, as instâncias abrangem uma ampla gama de casos de teste, permitindo a análise comparativa de diferentes abordagens e a avaliação do desempenho dos algoritmos em diferentes contextos. Por padrão, as instâncias seguem a seguinte estrutura:

ESTRUTURA DAS INSTÂNCIAS - EXEMPLO
❖ **NAME:** A-n32-k5;
Nome da instância, onde n = quantidade de nós do grafo/pontos no plano cartesiano e k= quantidade de caminhões disponíveis.
❖ **COMMENT:** (Augerat et al, No of trucks: 5, Optimal value: 784);

Comentário sobre a instância, contendo novamente o número de caminhões disponíveis e a solução ótima.

- **TYPE**: CVRP

Tipo do problema (roteamento de veículos).

- **DIMENSION** : 32

Dimensão da instância, ou seja, a quantidade de nós do grafo/pontos no plano cartesiano.

- **EDGE_WEIGHT_TYPE** : EUC_2D

Tipo de peso entre os nós do grafo, em nosso caso todas são EUC_2D, ou seja, o peso da aresta equivale a distância euclidiana entre os dois grafos.

- **DIMENSION** : 32

Dimensão da instância, ou seja, a quantidade de nós do grafo/pontos no plano cartesiano.

- **CAPACITY** : 100

Capacidade do veículo

- **NODE_COORD_SECTION**

Índice	X	Y
1	**18**	**76**
2	**96**	**44**
3	**50**	**5**
4	**49**	**8**
5	**13**	**7**

Consiste em 3 colunas, a primeira é o índice, servindo como referência. A segundo e a terceira colunas se referem às coordenadas no plano cartesiano. Essas, serão usadas para o cálculo da distância entre os clientes.

- **DEMAND_SECTION**

Índice	Demanda
1	0
2	19
3	21
4	6

Consiste em 2 colunas, a primeira se refere ao índice do ponto e a segunda a demanda necessária naquele nó. Em nossos exemplos o depósito se encontra sempre no índice 1, portanto a sua demanda será 0. Os demais são os clientes e suas demandas.

OBSERVAÇÕES:

As instâncias não restringem caminhos, então, concluímos que todos os caminhos estão interligados, formando um grafo completo.

3. ALGORITMOS

3.1 HEURÍSTICA CONSTRUTIVA SEMI-GULOSA

1. Receber o parâmetro k (percentual a ser selecionado);
2. Listar todos os clientes disponíveis para serem visitados;
3. Enquanto houver disponíveis:
   - 3.1. Definir a quantidade K de clientes baseado no parâmetro k;
   - 3.2. Definir a rota que tem a maior capacidade disponível;
   - 3.3. Cliente atual = último cliente da rota;
   - 3.4. Listar os disponíveis em ordem crescente de distância do cliente atual;
   - 3.5. Selecionar um cliente aleatório dentre os K primeiros da lista.;
   - 3.6. Adicionar na solução atual;
4. Retornar a solução.

3.2 ITERATED GREEDY

1. Receber os parâmetros M (máximo de iterações), k (percentual a ser selecionado), d(percentual a ser destruído);
2. Definir uma solução inicial aleatória, através da heurística semi-gulosa (definida no item 3.1) com k=100;
3. Enquanto o máximo de iterações M não for atingido;

3.1. Destruir d% dos elementos aleatoriamente entre todas as rotas;
3.2. Reconstruir as rotas através utilizando a heurística semi-gulosa, repassando k;
3.3. Se a nova solução atual for melhor que a melhor até o momento, essa passa a ser a melhor solução;

4. Retorna a melhor solução.

3.3 BUSCA LOCAL SIMPLES, COM ESTRATÉGIA DE SELEÇÃO MELHORIA ALEATÓRIA

1. Receber o parâmetro M (máximo de iterações);
2. Definir uma solução inicial aleatória, através da heurística semi-gulosa (definida no item 3.1) com k=100;
3. Enquanto o máximo de iterações M não for atingido ou a solução ótima ser encontrada:
    a. Construir uma lista com todas as soluções vizinhas da solução atual através de buscas locais, trocando todos elementos entre si (intra e inter rotas);
    b. Filtrar as soluções que geram uma solução melhor que a atual e que não violem a restrição de capacidade dos caminhões;
    c. Caso haja soluções disponíveis, selecionar aleatoriamente uma delas para ser a atual. Caso contrário, retornar a última solução;
4. Retorna a última solução.

3.4 BUSCA LOCAL ITERADA

1. Receber os parâmetros M (máximo de iterações) e n (número de perturbações);
2. Definir uma solução inicial aleatória, através da heurística semi-gulosa (definida no item 3.1) com k=100;
3. Enquanto o máximo de iterações M não for atingido ou a solução ótima ser encontrada:
    a. Executar a busca local simples (definida no item 3.3), na solução atual;
    b. Aplicar a perturbações na solução atual, trocando dois elementos aleatoriamente (intra ou inter rotas), por n vezes;
4. Retorna a última solução

3.4 GRASP + BUSCA LOCAL SIMPLES

1. Receber o parâmetro M (máximo de iterações), k (percentual a ser selecionado), I(número de iterações internas);
2. Enquanto o máximo de iterações M não for atingido ou a solução ótima ser encontrada:
    a. Definir uma solução atual semi gulosa (definida no item 3.1), repassando o parâmetro k para instância;
    b. Executar a busca local simples (definida no item 3.3), na solução atual, repassando o parâmetro I como máximo de iterações da instância;
    c. Verificar se a solução da busca local simples é melhor que a melhor solução;
3. Retorna a melhor solução

3.5 GRASP + BUSCA LOCAL ITERADA

1. Receber os parâmetro Ms (máximo de iterações), k (percentual a ser selecionado), n (número de perturbações) e I(número de iterações internas);
2. Enquanto o máximo de iterações M não for atingido ou a solução ótima ser encontrada:
    a. Definir uma solução atual semi gulosa (definida no item 3.1), repassando o parâmetro k para instância;
    b. Executar a busca local iterada (definida no item 3.3), na solução atual, repassando para a instância o parâmetro I como máximo de iterações e n como número de perturbações;
    c. Verificar se a solução da busca local iterada é melhor que a melhor solução;
3. Retorna a melhor solução.

3.5 GRASP + ITERATED GREEDY

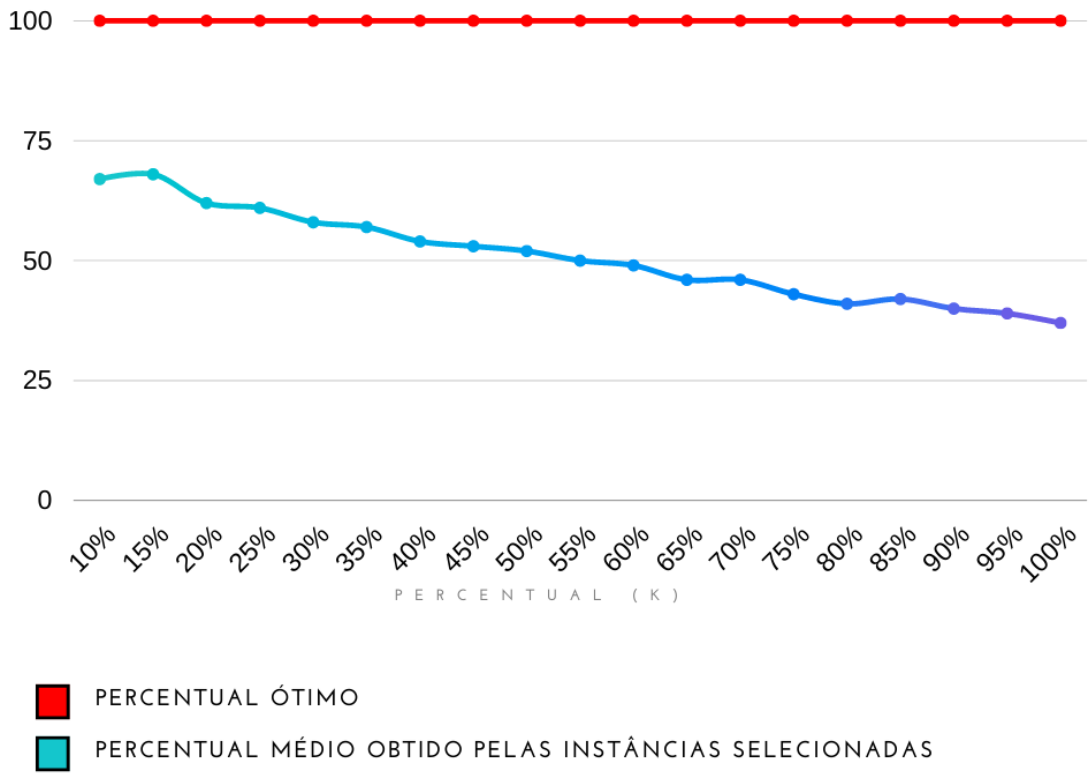
1. Receber os parâmetros M (máximo de iterações), k (percentual a ser selecionado), I(número de iterações internas) e D(percentual a ser destruído).;
2. Enquanto o máximo de iterações M não for atingido ou a solução ótima ser encontrada:
    a. Definir uma solução atual semi gulosa (3.1), repassando o parâmetro k para instância;
    b. Executar a busca gulosa iterada (definida no item 3.2), na solução atual, repassando o parâmetro I como máximo de iterações da instância, D como a destruição e k como construção;
    c. Verificar se a solução da busca gulosa iterada é melhor que a melhor solução
3. Retorna a melhor solução

4. RESULTADOS

Os resultados das análises do algoritmo foram obtidos com base nos melhores parâmetros encontrados após várias execuções com diferentes valores. Depois de identificar os melhores parâmetros, os algoritmos foram executados repetidamente para obter uma média das soluções. Por fim, a média das soluções encontradas com os melhores parâmetros é comparada com a solução ideal, fornecendo um percentual médio.

4.1 - ANÁLISE DO SEMI GREEDY

GRÁFICO DE EVOLUÇÃO - PERCENTUAL DE K



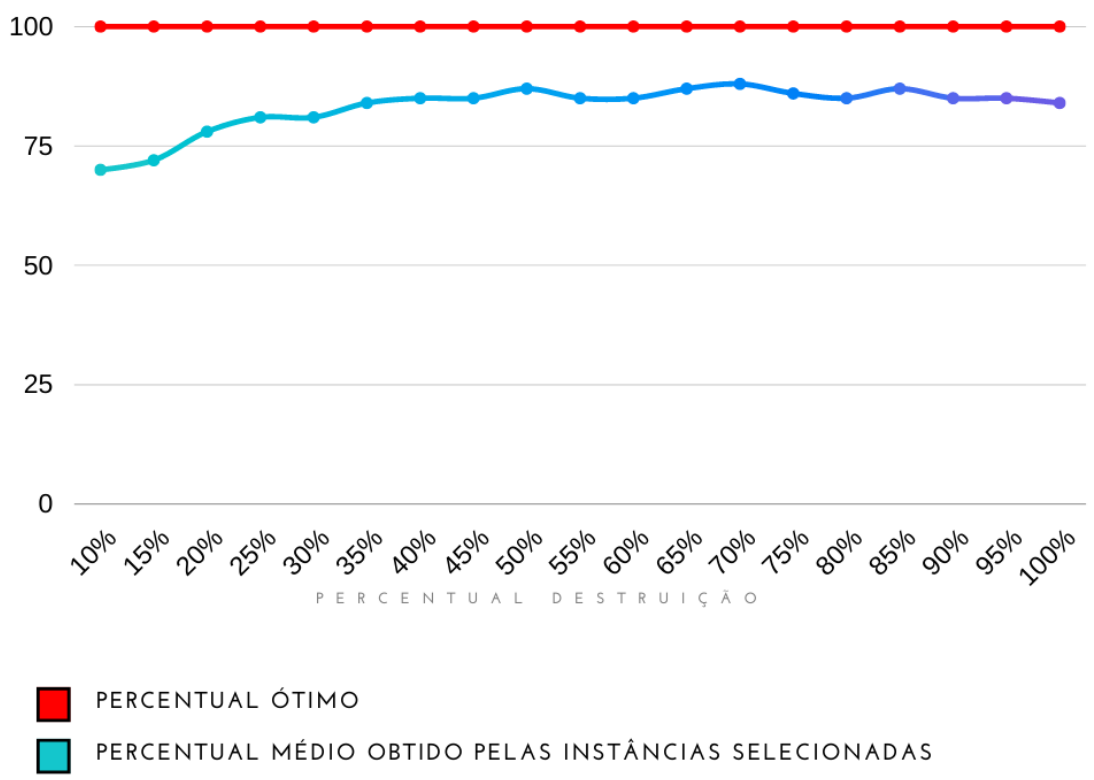
Melhor valor de parâmetro (k): 15%

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	1329.94	58%
instances/A/A-n33-k6.vrp	742	927.52	79%
instances/A/A-n37-k5.vrp	669	975.11	68%
instances/A/A-n39-k5.vrp	822	1082.89	75%

instances/A/A-n44-k6.vrp	937	1464.34	63%
instances/A/A-n45-k7.vrp	1146	1721.07	66%
instances/A/A-n46-k7.vrp	914	1614.57	56%
instances/A/A-n48-k7.vrp	1073	1811.96	59%
instances/A/A-n53-k7.vrp	1010	1543.65	65%
instances/A/A-n54-k7.vrp	1167	1677.42	69%
instances/A/A-n60-k9.vrp	1354	2051.96	65%
instances/A/A-n62-k8.vrp	1288	1829.94	70%
instances/A/A-n63-k10.vrp	1616	2327.93	69%
instances/A/A-n65-k9.vrp	1174	2166.17	54%
instances/A/A-n80-k10.vrp	1763	2622.5	67%

4.2 - ANÁLISE DO ITERATED GREEDY

GRÁFICO DE EVOLUÇÃO - PERCENTUAL DE D



Melhor valor de parâmetro (D): 70%

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	926.98	84%
instances/A/A-n33-k6.vrp	742	782.34	94%
instances/A/A-n37-k5.vrp	669	847.51	78%
instances/A/A-n39-k5.vrp	822	910.89	90%
instances/A/A-n44-k6.vrp	937	1345.72	69%
instances/A/A-n45-k7.vrp	1146	1381.53	82%
instances/A/A-n46-k7.vrp	914	1369.94	66%
instances/A/A-n48-k7.vrp	1073	1487.97	72%
instances/A/A-n53-k7.vrp	1010	1199.39	84%
instances/A/A-n54-k7.vrp	1167	1258.50	92%
instances/A/A-n60-k9.vrp	1354	1608.81	84%

instances/A/A-n62-k8.vrp	1288	1505.13	85%
instances/A/A-n63-k10.vrp	1616	1845.68	87%
instances/A/A-n65-k9.vrp	1174	1525.56	76%
instances/A/A-n80-k10.vrp	1763	1923.02	92%

- Após uma análise da execução do algoritmo Iterated Greedy em relação à solução ótima, constatamos que, em certa medida, o aumento do desempenho está ligado aos custos elevados da solução inicial gerada pelo Semi-Greedy, pois ao destruirmos a solução obtemos resultados melhores e conforme avançamos o percentual estabilizamos o custo médio das soluções obtidas. Além disso, observamos um significativo aumento no tempo de execução do algoritmo devido à utilização de um percentual elevado de destruição.

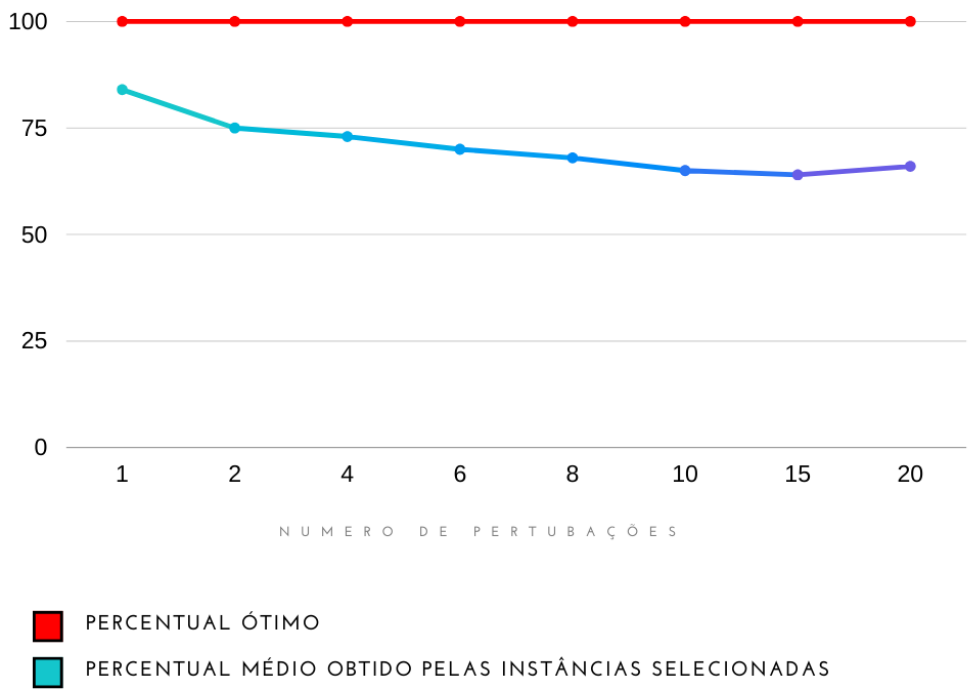
4.3 - ANÁLISE DO SIMPLE LOCAL SEARCH

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	999,74	78%
instances/A/A-n33-k6.vrp	742	758,7	97%
instances/A/A-n37-k5.vrp	669	874,35	76%
instances/A/A-n39-k5.vrp	822	894,26	91%
instances/A/A-n44-k6.vrp	937	1278,76	73%
instances/A/A-n45-k7.vrp	1146	1303,06	87%
instances/A/A-n46-k7.vrp	914	1340,37	68%
instances/A/A-n48-k7.vrp	1073	1455,27	73%
instances/A/A-n53-k7.vrp	1010	1197,18	84%
instances/A/A-n54-k7.vrp	1167	1376,99	84%
instances/A/A-n60-k9.vrp	1354	1548,09	87%
instances/A/A-n62-k8.vrp	1288	1703,33	75%
instances/A/A-n63-k10.vrp	1616	2079,3	77%

instances/A/A-n65-k9.vrp	1174	1968,26	59%
instances/A/A-n80-k10.vrp	1763	2222,71	79%

4.4 - ANÁLISE DO ITERATED LOCAL SEARCH

GRÁFICO DE EVOLUÇÃO - VALOR DE M



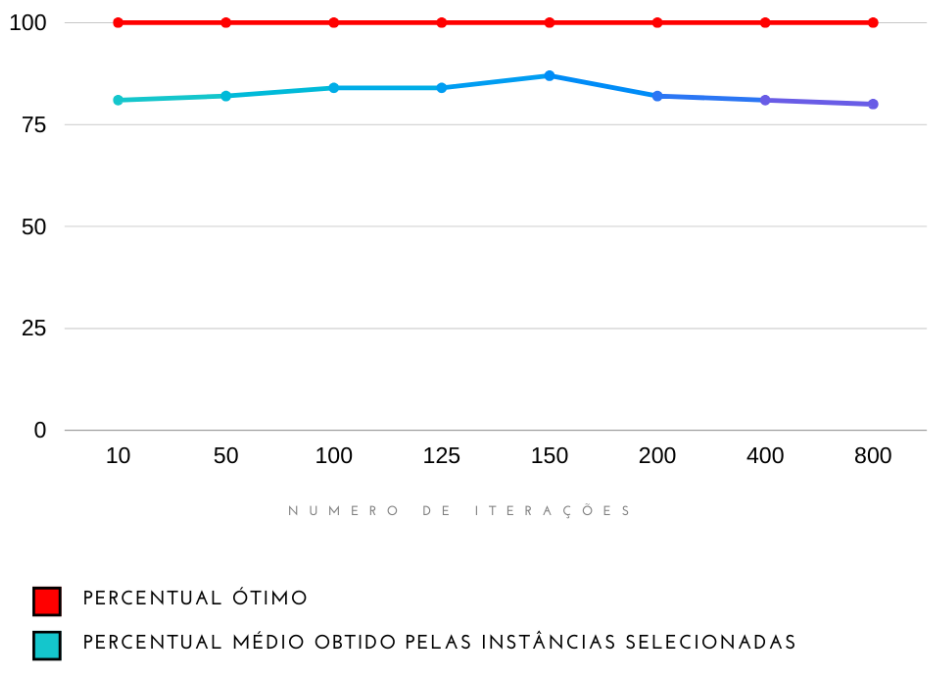
Melhor valor de parâmetro (m): 1

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	998.46	78%
instances/A/A-n33-k6.vrp	742	845.22	87%
instances/A/A-n37-k5.vrp	669	773.66	86%
instances/A/A-n39-k5.vrp	822	913.15	90%
instances/A/A-n44-k6.vrp	937	1272.28	73%
instances/A/A-n45-k7.vrp	1146	1493.18	76%
instances/A/A-n46-k7.vrp	914	1278.48	71%

instances/A/A-n48-k7.vrp	1073	1445.23	74%
instances/A/A-n53-k7.vrp	1010	1181.49	85%
instances/A/A-n54-k7.vrp	1167	1399.66	83%
instances/A/A-n60-k9.vrp	1354	1453.82	93%
instances/A/A-n62-k8.vrp	1288	1239.17	96%
instances/A/A-n63-k10.vrp	1616	1820.95	88%
instances/A/A-n65-k9.vrp	1174	1582.53	74%
instances/A/A-n80-k10.vrp	1763	1817.43	97%

4.5 - ANÁLISE GRASP + SIMPLE LOCAL SEARCH

GRÁFICO DE EVOLUÇÃO - VALOR DE I



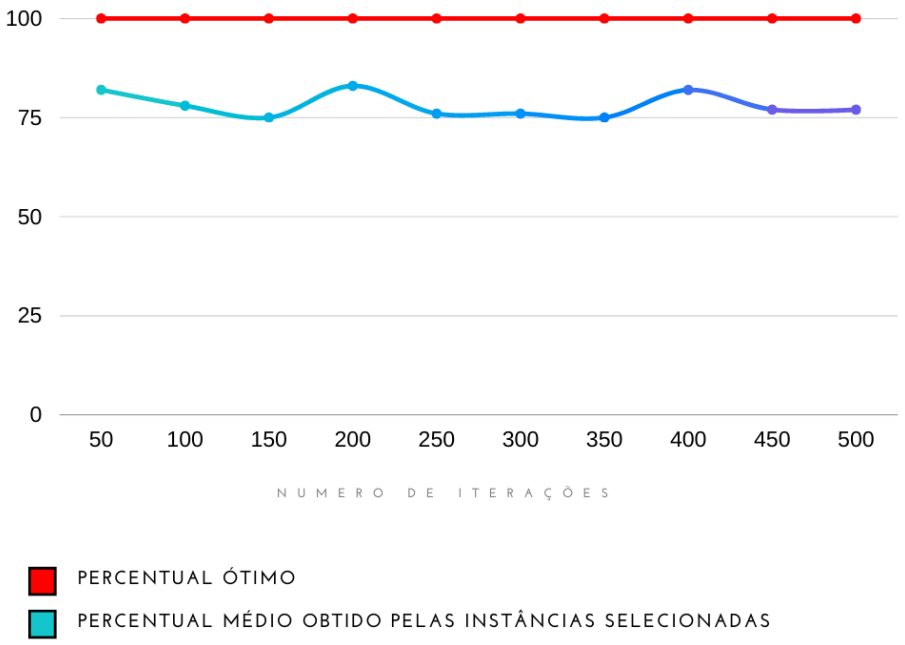
Melhor valor de parâmetro (I): 150

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	915.83	85%

instances/A/A-n33-k6.vrp	742	779.41	95%
instances/A/A-n37-k5.vrp	669	767.28	87%
instances/A/A-n39-k5.vrp	822	855.62	96%
instances/A/A-n44-k6.vrp	937	1220.42	76%
instances/A/A-n45-k7.vrp	1146	1338.87	85%
instances/A/A-n46-k7.vrp	914	1355.46	67%
instances/A/A-n48-k7.vrp	1073	1367.59	78%
instances/A/A-n53-k7.vrp	1010	1100.46	91%
instances/A/A-n54-k7.vrp	1167	1277.22	91%
instances/A/A-n60-k9.vrp	1354	1427.27	94%
instances/A/A-n62-k8.vrp	1288	1427.59	90%
instances/A/A-n63-k10.vrp	1616	1730.93	93%
instances/A/A-n65-k9.vrp	1174	1640.42	71%
instances/A/A-n80-k10.vrp	1763	1923.27	91%

4.6 - ANÁLISE GRASP + ITERATED LOCAL SEARCH

GRÁFICO DE EVOLUÇÃO - VALOR DE I



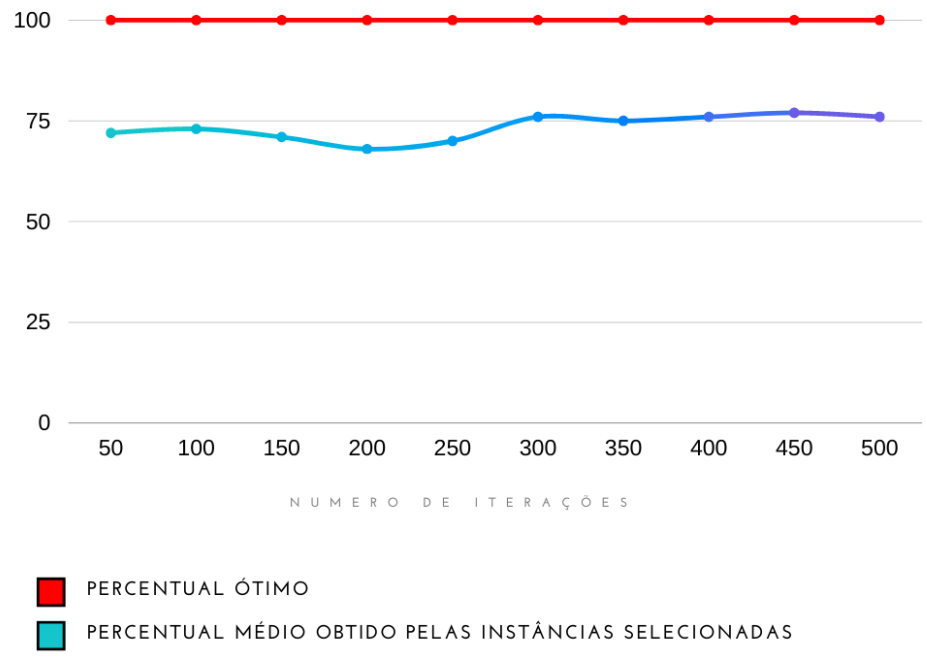
Melhor valor de parâmetro (l): 200

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	867.43	90%
instances/A/A-n33-k6.vrp	742	758.82	97%
instances/A/A-n37-k5.vrp	669	797.76	83%
instances/A/A-n39-k5.vrp	822	834.4	98%
instances/A/A-n44-k6.vrp	937	1307.23	71%
instances/A/A-n45-k7.vrp	1146	1303.81	87%
instances/A/A-n46-k7.vrp	914	1369.69	66%
instances/A/A-n48-k7.vrp	1073	1495.39	71%
instances/A/A-n53-k7.vrp	1010	1281.61	78%
instances/A/A-n54-k7.vrp	1167	1636.66	71%
instances/A/A-n60-k9.vrp	1354	1619.02	83%
instances/A/A-n62-k8.vrp	1288	1345.45	95%
instances/A/A-n63-k10.vrp	1616	2002.35	80%

instances/A/A-n65-k9.vrp	1174	1358.36	86%
instances/A/A-n80-k10.vrp	1763	1956.78	90%

4.7 - ANÁLISE GRASP + ITERATED GREEDY

GRÁFICO DE EVOLUÇÃO - VALOR DE I



Melhor valor de parâmetro (I): 300

INSTÂNCIAS	SOLUÇÃO ÓTIMA (a)	SOLUÇÃO OBTIDA (b)	%(a/b)
instances/A/A-n32-k5.vrp	784	1555.69	50%
instances/A/A-n33-k6.vrp	742	1103.26	75%
instances/A/A-n37-k5.vrp	669	1103.26	60%
instances/A/A-n39-k5.vrp	822	1170.36	81%
instances/A/A-n44-k6.vrp	937	1548.01	80%
instances/A/A-n45-k7.vrp	1146	1525.12	75%
instances/A/A-n46-k7.vrp	914	1441.89	63%
instances/A/A-n48-k7.vrp	1073	1740.51	61%

instances/A/A-n53-k7.vrp	1010	1444.96	69%
instances/A/A-n54-k7.vrp	1167	1609.05	63%
instances/A/A-n60-k9.vrp	1354	1728.03	78%
instances/A/A-n62-k8.vrp	1288	1826.14	70%
instances/A/A-n63-k10.vrp	1616	2149.76	75%
instances/A/A-n65-k9.vrp	1174	1923.29	61%
instances/A/A-n80-k10.vrp	1763	2683.43	65%

4.8 - CONCLUSÃO DA ANÁLISE GERAL

No geral, todos os algoritmos apresentaram um desempenho satisfatório em relação a todas as instâncias, independentemente do seu tamanho. É importante destacar que, ao serem utilizados com GRASP, houve uma notável redução na velocidade de execução de todos os algoritmos, porém, ainda assim, eles conseguiram alcançar resultados aceitáveis em termos de custo.

Destacamos o excelente desempenho do algoritmo Iterated Greedy, que obteve o melhor custo médio em comparação aos outros algoritmos implementados. No entanto, é importante mencionar que, ao ser utilizado em conjunto com o GRASP, houve uma drástica queda no seu desempenho.

5. CONCLUSÃO

Com base nos resultados obtidos, podemos concluir que o estudo realizado apresentou consistência em relação aos objetivos propostos. No entanto, identificamos áreas que requerem melhorias em termos de código, visando encontrar soluções mais eficientes e que demandem menos tempo de processamento.

A primeira área de melhoria é relacionada ao método de definição da rota atual na heurística semi-gulosa. Atualmente, utiliza-se a rota com a maior capacidade disponível. Recomenda-se modificar esse método, explorando outras estratégias que possam resultar em rotas mais eficientes.

Outra área que necessita de aprimoramento é o método de destruição das rotas na heurística gulosa iterada. No método atual, os elementos são destruídos de forma aleatória, e os elementos restantes são utilizados para formar novas rotas completas antes da reconstrução. Sugere-se modificar esse método, buscando uma abordagem mais eficiente para a destruição e reconstrução das rotas.

Além disso, é importante considerar a melhoria do método de montagem das rotas vizinhas na busca local. É recomendado realizar modificações nesse método a

fim de evitar iterações desnecessárias, otimizando a velocidade do algoritmo e facilitando a busca por melhores parâmetros.

Em suma, embora os resultados encontrados tenham sido consistentes, é crucial implementar as melhorias propostas no código para alcançar soluções mais eficientes e reduzir o tempo necessário para execução das heurísticas, assim, possibilitando uma melhor análise de parâmetros