LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 3 a 5 de novembro de 2020

# APLICAÇÃO DA BUSCA TABU PARA SOLUÇÃO DO PROBLEMA DE ROTEAMENTO DE VEÍCULOS CAPACITADO

**Lucas Caldeira de Oliveira, Bruno Duarte, Marcelo Teixeira, Marco Antonio Barbosa**
Universidade Tecnológica Federal do Paraná
Via do Conhecimento, s/n - km 01, Pato Branco - PR
{lucasoliveira.2017,brunoduarte}@alunos.utfpr.edu.br,
{marceloteixeira, mbarbosa}@utfpr.edu.br

**RESUMO**

O Problema de Roteamento de Veículos Capacitado (PRVC) está presente em diversas situações do cotidiano relacionadas ao transporte de bens e serviços. Qualquer economia no custo associado a esses problemas geraria uma redução direta de gastos e impactos ambientais positivos. Dada a grande relevância do PRVC, propõe-se sua resolução por meio da Busca Tabu, uma metaheurística amplamente utilizada devido a sua alta eficácia e baixa complexidade em problemas dessa classe. Neste trabalho são desenvolvidas três implementações variantes do método: duas determinísticas e uma estocástica. O teste e a validação utilizam instâncias de teste presentes na literatura e, ao final, é feita uma discussão sobre as características de cada implementação variante, assim como uma comparação com outros trabalhos recentes, mostrando que os métodos propostos são competitivos.

**PALAVRAS CHAVE. Busca Tabu, Problema de Roteamento de Veículos, Metaheurísticas.**

**MH - Metaheurísticas, OC - Otimização Combinatória, L&T - Logística e Transportes.**

**ABSTRACT**

The Capacitated Vehicle Routing Problem (CVRP) is present in several everyday situations related to the transport of goods and services. Any cost savings associated with these problems would generate a direct reduction in spending and positive environmental impacts. Given the great relevance of CVRP, its resolution is proposed through Tabu Search, a metaheuristic widely used due to its high effectiveness and low complexity in problems from this class. In this work are developed three variant implementations of the method: two deterministic and one stochastic. The test and the validation use test instances common in the literature and, in the end, a discussion is made about the characteristics of each variant implementation, as well as a comparison with other recent works, showing that the proposed methods are competitive.

**KEYWORDS. Tabu Search, Vehicle Routing Problem, Metaheuristics.**

**MH - Metaheuristics, CO - Combinatorial Optimization, L&T - Logistics and Transport.**

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

## 1. Introdução

O Problema de Roteamento de Veículos (PRV ou, do inglês, VRP - *Vehicle Routing Problem*) é um clássico problema de otimização combinatória derivado do Problema do Caixeiro Viajante. O problema, originalmente proposto, consiste em encontrar o conjunto ótimo de rotas a serem percorridas por uma frota de veículos de forma que supram a demanda de um conjunto de clientes, tal que cada cliente deve ser visitado uma única vez, todos os clientes devem ser visitados, e os veículos devem partir e retornar a um depósito predefinido [Toth e Vigo, 2001].

Desde quando foi proposto por Dantzig e Ramser [1959], o Problema de Roteamento de Veículos vem despertando o interesse de diversos pesquisadores devido à sua grande relevância socioeconômica, visto que possui uma relação direta com os setores de logística e transporte de bens e serviços, alinhada com a possibilidade de redução significativa no custo do transporte e no impacto ambiental [Ibrahim et al., 2019; Faulin et al., 2011; Simas, 2007].

A definição clássica do problema vem apresentando variações ao longo dos anos, de acordo com novas restrições que são acrescidas ao problema original. São exemplos destas variações o PRV com Frota Heterogênea, com Janela de Tempo, com Entrega e Coleta, dentre vários outros [Toth e Vigo, 2001; Golden et al., 2008].

O presente artigo aborda o PRV Capacitado (PRVC) com Frota Homogênea, isto é, os veículos apresentam um limite de carga uniforme. Portanto, a melhor solução é aquela que minimiza o custo das rotas – definido como a distância percorrida –, utilizando a menor quantidade de veículos possível para atender à demanda dos clientes [Laporte, 1992].

Problemas da natureza dos PRVs, que buscam encontrar a solução ótima dentre o espaço de soluções existentes, são ditos NP-Difíceis. Solucionar tais problemas por meio de buscas exaustivas conduzem a um tempo de execução exponencial, pois encontrar a solução ótima se traduz em conhecer e testar todas as possíveis soluções para uma dada instância do problema [Papadimitriou e Steiglitz, 1982] e este, por sua vez, na prática, pode conter muitas instâncias, tornando a solução inviável de ser obtida por método exatos.

Sendo a resolução por métodos exaustivos ineficiente para instâncias suficientemente grandes, uma alternativa de solução é por meio do uso de metaheurísticas [Gendreau e Potvin, 2010]. Tais métodos retornam, em tempo polinomial, uma solução viável ao problema, muitas vezes se aproximando satisfatoriamente ou até mesmo atingindo o resultado ótimo [Talbi, 2009]. Estes métodos são mais rápidos pois reduzem o espaço de busca, e em função disso perde-se a garantia de atingir a solução ótima. São exemplos de metaheurísticas: GRASP, *Simulated Annealing*, Algoritmos Genéticos, Colônia de Formigas, Busca Tabu, dentre outras [Gendreau e Potvin, 2010].

Criada por Glover em 1986, a Busca Tabu (BT) vem sendo amplamente adotada desde então na resolução de problemas de otimização, tanto por sua eficácia quanto pela simplicidade de implementação [Gendreau, 2003]. A BT difere-se de heurísticas tradicionais, como por exemplo a Descida/Subida de Encosta, pela possibilidade de percorrer soluções vizinhas que não sejam soluções de melhora. Tal estratégia é realizada como mecanismo de escape de ótimos locais.

A BT utiliza uma memória adaptativa, denominada Lista Tabu, que registra os movimentos efetuados e evita que a busca se restrinja a regiões de ótimo local ao possibilitar a movimentação para vizinhos piores e ao aplicar mecanismos anti-ciclagem que impedem o retorno a uma solução já previamente encontrada [Glover et al., 1993].

São propostas três implementações da Busca Tabu – duas determinísticas e uma estocástica – e algumas funções adicionais, que são abordadas na seção 3. Os métodos foram testados utilizando 74 instâncias de Augerat et al. [1998] e a análise contida na seção 4 considera fatores como desempenho médio, tempo de execução e qualidade das soluções encontradas, de onde pode-

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings

se adiantar que os métodos, em conjunto, foram bem-sucedidos em 51 dos casos, com um erro médio de 0,8% ao considerar o melhor resultado para cada instância em todo o conjunto testado.

## 2. Formalização do Problema

Tanto o PRV quanto suas variações podem ser modelados por um grafo $G(V, E)$, no qual cada vértice $v \in V$ representa um cliente com uma demanda $q$ a ser atendida, e cada aresta $e \in E$ representa um caminho bidirecional entre 2 vértices com um custo $c$ associado. Uma solução possível é aquela que, partindo do vértice de origem $v_0$ (depósito), passa uma única vez por cada um dos demais vértices e retorna à origem [Ibrahim et al., 2019; Ralphs et al., 2003].

O PRVC adiciona ao problema original, PRV, a restrição do limite de carga nos veículos, de forma que se busca o conjunto mínimo de veículos que atenda às demandas no menor custo de rota. Trata-se, portanto, da minimização de custos. As condições que compõem o enunciado são:

- o grafo $G(V, E)$ é completo e simétrico, possui um único depósito $v_0$, e inclui $n = |V \setminus \{v_0\}|$ vértices com demandas a serem atendidas;
- cada vértice $v_i \in V \setminus \{v_0\}$ corresponde a um cliente com uma demanda $q_i > 0$, e somente o depósito $v_0$ possui demanda nula ($q_0 = 0$);
- cada aresta $e_{i,j}$ mapeia um caminho direcionado entre dois vértices $(v_i, v_j)$ e contém um custo associado $c_{i,j}$;
- o vértice de início e fim de uma rota $R$ deve sempre ser o depósito (ciclo fechado), sendo $R$ um subgrafo assimétrico de $G$ no qual todos os vértices tem apenas 1 aresta entrando e 1 saindo e que possui $v_0$, obrigatoriamente.
- a demanda $q_i$ de cada cliente deve ser satisfeita em uma única entrega, e cada vértice só pode ser visitado uma única vez, exceto o depósito;
- a entrega é feita por veículos homogêneos com limite de carga $Q \in \mathbb{N}^*$, de forma que a soma das demandas dos vértices associados a uma rota $R$ não supere $Q$.

Define-se uma solução válida $S$ como sendo uma união de $k$ rotas (*tours*) $R_1, R_2, \cdots, R_k$ em $G$ [Ralphs et al., 2003] que atendam a cada vértice conforme enunciado acima e de forma que a intersecção de quaisquer rotas seja o depósito. Portanto, uma possível representação de $S$ é

$$S = R_1 \cup \cdots \cup R_k \subset G(V, E) \;\mid\; \forall R,\ \sum_{i=1}^{|R|} q_i \leq Q \;\wedge\; R_i \cap R_j = \{v_0\} \cdot \tag{1}$$

O custo $c_{i,j}$ de cada aresta depende unicamente do deslocamento necessário para ir de $v_i$ a $v_j$ e é calculado através do arredondamento da distância euclidiana entre os vértices envolvidos. Dados dois vértices $v_i, v_j \in V$ com coordenadas cartesianas $(x, y)$, o custo da aresta associada é

$$c_{i,j} = dist(v_i, v_j) = \left\lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 0,5 \right\rfloor \cdot \tag{2}$$

Considerando uma solução $S = R_1 \cup \cdots \cup R_k$, o custo total da solução é a soma do custo de cada aresta percorrida, incluindo o custo para sair e retornar ao depósito de origem $v_0$. Denotando um vértice pertencente a uma rota $R_p$ como $v^p$, tem-se

$$c_{total}(S) = \sum_{p=1}^{|S|} \left( dist(v_0, v_1^p) + \sum_{i=2}^{|R_p|} dist(v_{i-1}^p, v_i^p) + dist(v_{|R_p|}^p, v_0) \right) , \tag{3}$$

ou, ainda, definindo-se $v_0^p = v_{|R_p|+1}^p = v_0$, é possível simplificar a Eq. (3) como

$$c_{total}(S) = \sum_{p=1}^{|S|} \sum_{i=0}^{|R_p|} dist(v_i^p, v_{i+1}^p) \cdot \tag{4}$$

O problema consiste, portanto, numa questão de minimização de custo, tal que a solução ótima $S^{\#}$ é aquela que possui o menor custo possível dentre todo o espaço de soluções $N(S)$, e

$$S^{\#} = min\ N(S) \ \Leftrightarrow\ c_{total}(S^{\#}) = min\left\{c_{total}(S) \mid S \in N(S)\right\} \cdot \tag{5}$$

Por se tratar de um problema da classe NP-difícil, encontrar $S^{\#}$ implica em verificar o custo de todas as $n!$ combinações de rota possíveis dentro do espaço $N(S)$ e, mesmo para uma quantidade pequena de vértices, tal abordagem exaustiva pode ser inviável. Para efeitos de ilustração, uma situação com $n = 20$ admitiria $2,432902 \cdot 10^{18}$ arranjos de vértices, enquanto que as instâncias testadas nesse trabalho possuem de 15 à 100 vértices além do depósito.

A alternativa de resolução trazida pelas heurísticas consiste em limitar a procura por uma ou mais vizinhanças de uma solução $N^*(S) \subset N(S)$, ao invés de fazê-la em todo o espaço de soluções, o que é plausível uma vez que $|N^*| \ll |N|$. No entanto, embora seja mais rápido concentrar a busca em apenas uma fração de N(S), essa alternativa impõe o risco de se vincular a solução à regiões de ótimo local, ou seja, ainda que se encontre uma solução $S^* = min\ N^*(S)$, é possível que $c_{total}(S^*) > c_{total}(S^{\#})$.

A fim de extrapolar a busca para fora dessas regiões de ótimo local, aproximando assim $S^*$ do ótimo global $S^{\#}$, podem ser adotadas técnicas adicionais de movimentação. A estratégia geral de busca, com a adição de um ou mais métodos de fuga de ótimos locais, é o que caracteriza as metaheurísticas [Gendreau e Potvin, 2010]. A explicação a respeito da escolha da Busca Tabu e a estruturação do método de resolução é dado a seguir.

## 3. Desenvolvimento da Solução Proposta

Muitas das metaheurísticas existentes se baseiam em fenômenos físicos – como o *Simulated Annealing*, análogo ao recozimento de metais – e biológicos – como os Algoritmos Genéticos, inspirados nos processos de seleção natural e mutação [Gendreau, 2003]. Dentre elas, uma se destaca na resolução de problemas como o PRVC, a *Busca Tabu* (BT), que tem se mostrado eficaz para encontrar soluções de boa qualidade para casos de teste clássicos na literatura [Simas, 2007].

A metaheurística BT se caracteriza por registrar os movimentos feitos na troca de uma solução corrente, $S_i$, para uma próxima solução $S_{i+1}$, durante a busca em sua vizinhança $N^*(S_i)$, tentando evitar que o mesmo movimento seja feito nas próximas $T$ iterações. A transição $S_i \rightarrow S_{i+1}$ ocorre escolhendo-se o vizinho com menor custo total dentro da vizinhança $N^*(S_i)$. Idealmente, busca-se uma transição tal que $c_{total}(S_{i+1}) < c_{total}(S_i)$, o que nem sempre é possível e implica em estratégias para contornar a estagnação em torno de ótimos locais. Desta forma, faz-se necessário definir as características gerais que regem a BT, desde a solução inicial até as estruturas internas de busca e controle, o que é discutido a seguir [Glover et al., 1993].

### 3.1. Detalhes da Implementação

Visando otimizar a utilização de memória e processamento, o grafo $G(V, E)$ foi implementado na forma de uma matriz de conectividade $C_{m \times m} = (c_{i,j})$, na qual $m = |V| = n + 1$ e cada elemento $c_{i,j}$ da matriz armazena o custo $dist(v_i, v_j)$.

Quanto à representação de uma solução qualquer $S$, utilizou-se uma estrutura contendo uma variável para o custo total e duas listas estáticas para o roteamento. A primeira lista armazena

a sequência de vértices na ordem em que são visitados, sem incluir passagens pelo depósito, e a segunda para registrar os índices da primeira lista que correspondem a inícios de rota. Essa estrutura facilita a execução de permutações na geração de soluções vizinhas e minimiza o custo de operações de inserção e remoção na lista maior, oriundos da mudança da posição de $v_0$ na sequência de vértices à medida que o arranjo da solução se altera.

Por exemplo, considere uma instância com nove vértices $v_1, v_2, ..., v_9$ mais o depósito $v_0$ ($m = 10$). Uma solução $S = R_1 \cup R_2 \cup R_3$ composta pelas rotas $R_1 : v_0 \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_8 \rightarrow v_0$, $R_2 : v_0 \rightarrow v_2 \rightarrow v_7 \rightarrow v_4 \rightarrow v_0$ e $R_3 : v_0 \rightarrow v_6 \rightarrow v_0$ representa os vértices sendo percorridos na sequência $0, 1, 3, 5, 8, 0, 2, 7, 4, 0, 6, 0$. A representação equivalente a essa solução, conforme proposto, é

```
S = {route, split, cost}
S.route = {1  3  5  8  2  7  4  6}
S.split = {0  4  7}
S.cost = c_total(S)
```

em que os vértices `S.route[0] = 1`, `S.route[4] = 2` e `S.route[7] = 6`, apontados pelos índices armazenados no vetor `S.split`, referem-se aos início de cada uma das três rotas.

A estratégia geral de resolução adotada neste artigo foi formulada em três etapas, facilitando assim uma análise comparativa do resultado computado em cada uma delas: (i) criação de uma solução inicial $S_0$; (ii) refinamento preliminar de $S_0$ usando um conjunto pré-definido de vizinhanças; (iii) aplicação da metaheurística partindo da solução resultante do passo anterior, usando a mesma seleção de vizinhanças e variando os parâmetros de entrada. Cada uma das etapas é detalhada na sequência.

### 3.2. Heurística de Construção

Para a geração de uma solução inicial $S_0$ foi empregado o *Método Guloso*, que tem por característica construir rapidamente uma solução viável para o problema. A estratégia gulosa adotada é a do *método do vizinho mais próximo*, que seleciona um vértice de partida (sempre o depósito) para iniciar a lista de vértices, e cada iteração seleciona o vértice não visitado mais próximo da posição atual (último vértice adicionado à lista) [Cunha et al., 2002]. Adicionalmente, nesse trabalho, à medida em que a lista de vértices é preenchida, incrementa-se também um contador para representar a ocupação do veículo. Quando não for mais possível inserir um novo vértice sem extrapolar o limite de carga $Q$, retorna-se para o depósito $v_0$ e uma nova rota é iniciada. O processo prossegue até que todos os vértices $v \in V$ tenham sido visitados.

### 3.3. Heurística de Refinamento

O método de refinamento adotado neste artigo é a *Busca Local* (BL), que consiste em percorrer de forma exaustiva uma vizinhança $N^*$ e obter o melhor vizinho $S^* = min\ N^*(S_0) \cup \{S_0\}$, de acordo com o critério da função objetivo adotada. Uma vizinhança pode ser definida como um espaço de soluções alcançáveis por meio de uma função de mapeamento $N^* : S_0 \rightarrow S'$ [Gendreau, 2003]. Qualquer vizinho $S'$ é mapeado realizando um movimento característico da vizinhança sobre a solução semente $S_0$. Mais de uma vizinhança pode ser adotada para aumentar o espaço de busca e possibilitar atingir soluções mais distantes de $S_0$, mas quanto maior a vizinhança e mais complexa a sua função de mapeamento, maior o tempo de busca.

Foram implementadas as funções de mapeamento *2-opt* e *swap* [Gendreau, 2003] para serem usadas como vizinhanças na BL e na BT.

### 3.4. Metaheurística

Proposta por Glover et al. [1993], a Busca Tabu é uma metaheurística que baseia-se em efetuar buscas locais e armazenar os movimentos realizados em estruturas de memória, para avançar no espaço de soluções e evitar ciclagem em torno de ótimos locais. Por ser de simples entendimento e ter se mostrado eficiente na resolução de problemas derivados do PRV [Dantzig e Ramser, 1959], muitos trabalhos classificam esse método como um dos que apresentam os melhores resultados nesse domínio de problema [Simas, 2007; Gendreau, 2003].

O destaque do algoritmo são as estruturas de memória, que Glover et al. [1993] classifica em dois tipos: memória de curto prazo, ou lista tabu, que registra os últimos movimentos efetivos (que levaram à uma solução melhor que o atual) e os restringe por $T$ iterações; e memória de longo prazo, também chamado de lista de frequência, cujo intuito é direcionar a busca à determinadas regiões da vizinhança com base na análise da frequência dos movimentos.

Os movimentos registrados na memória de curto prazo são denominados *movimentos tabu*, e a maneira de lidar com eles – seja no armazenamento, tempo de restrição ou tamanho da memória – é crucial para o desempenho da BT, por isso esse é um dos fatores genéricos no pseudo-código e cabe ao projetista definir como será implantado. Já a memória de longo prazo funciona como um mecanismo de direcionamento de busca em determinadas fases de execução, e por não ser central no método de resolução sua inclusão é opcional.

Outra parte fundamental da Busca Tabu é a BL, que opera controlada por dois critérios: de restrição e de aspiração. Esses critérios consistem, sucintamente, em regras para determinar quando uma solução vizinha será ou não será escolhida. Enquanto o critério de restrição opera impedindo a escolha de uma solução, se essa estiver sido gerada por um movimento-tabu, ou impondo penalidades – baseado nas estruturas de memória de curto e longo prazo –, o critério de aspiração oferece um relaxamento, permitindo que as restrições sejam ignoradas sob certas circunstâncias e movimentos-tabu sejam aceitos [Glover et al., 1993; Gendreau, 2003].

Existem diversas variações da BT presentes na literatura, incluindo versões determinísticas, estocásticas, métodos híbridos, adaptativos, etc. Com o objetivo de testar a eficiência da metaheurística sem se limitar a uma única versão, foram implementadas três variantes do algoritmo genérico proposto por Glover et al. [1993]. De modo geral, as três variantes compartilham da mesma estrutura e as características principais são detalhadas a seguir.

**Estruturas fundamentais:** o algoritmo opera manipulando 4 soluções – vizinho $S_{ngbr}$, solução corrente/semente $S_{curr}$, próxima solução semente $S_{next}$ e melhor solução encontrada $S_{best}$ – e requer uma solução de partida $S_0$ para inicializar $S_{best}$ e $S_{curr}$. Ele também requer a seleção de um conjunto de vizinhanças $\{N^*\}$ e a definição dos parâmetros $\langle T, K_{max}\rangle$ – período de restrição e número máximo de iterações sem melhora, respectivamente;

**Condição de movimento:** um vizinho $S_{ngbr} \in N_i^*(S_{curr})$ que seja melhor que $S_{next}$ será selecionado para ser a próxima semente de busca, sendo que a mudança só será efetivada após a conclusão do ciclo atual de buscas;

**Ciclo de buscas:** um ciclo de buscas é uma sequência de buscas locais dentro de $\{N^*\}$. Empiricamente, obteve-se um melhor resultado ao permitir que $S_{curr}$ fosse atualizada com $S_{next}$ a cada troca de vizinhança $N_i^* \rightarrow N_{i+1}^*$ (semelhante à estratégia do VNS), enquanto que o registro do movimento na lista-tabu só ocorre após o fim do ciclo;

**Critério de restrição (Movimento tabu):** um movimento derivado de permutações nos índices $i, j$, que tenha sido efetivado, será registrado na lista-tabu e ficará proibido durante os próximos $T$ ciclos de busca. Assim, um vizinho $S_{ngbr}$ de custo total associado menor

que o custo de $S_{curr}$ será rejeitado caso o movimento que o gerou esteja proibido;

**Critérios de aspiração:** uma solução $S_{ngbr}$ que seja melhor que $S_{best}$ será sempre aceita, independente do critério de restrição. Caso o ciclo de buscas tenha acabado de começar e $S_{next}$ não tenha sido inicializado, ele receberá o primeiro vizinho alcançável, novamente ignorando o critério de restrição;

**Atualização da lista-tabu:** ao final do ciclo de buscas, os índices $i, j$ da lista referentes ao último movimento efetivo serão avaliados com o valor $T$, indicando que qualquer movimentação nesses índices estará sujeita ao critério de restrição. Os demais índices são decrementados até que se atinja o valor 0, liberando-os da restrição;

**Condição de parada:** o ciclo de buscas será executado sistematicamente até que se complete $K_{max}$ iterações sem haver melhora em $c_{total}(S_{best})$. Para isso, um contador $K$ é incrementado ao final de cada ciclo e zerado sempre que $S_{best}$ recebe um novo valor.

Sobre a marcação dos índices na lista-tabu, tanto as vizinhanças *swap* e *2-opt* utilizam dois índices para definir a região de troca e somente esses índices entram na lista. Para a vizinhança *swap*, esse tipo de lógica é intuitiva, visto que a função de mapeamento permuta apenas os índices referentes. Já para vizinhança *n-opt*, os índices representam delimitadores da região de trocas e parece contra-intuitivo registrar apenas os índices das extremidades. Ainda assim, esse tipo de abordagem é menos restritiva e permite que o algoritmo avance sem ficar impedido pela ausência de movimentos permitidos.

É importante ressaltar que as decisões de implementação adotadas neste trabalho diferem dos algoritmos-padrão detalhados em Glover et al. [1993] e Gendreau [2003], apesar de esta ser uma característica de todas as metaheurísticas cujo pseudo-código represente genericamente o método, não expondo a sequência de passos a ser estritamente seguida. A escolha de possibilitar saltos parciais entre cada i-ésima vizinhança ao atualizar $S^{i+1}_{curr} \leftarrow S^{i}_{next}$ e registrar apenas o último salto – que indiretamente representa o movimento que levou mais longe –, assim como a inicialização de $S^{1}_{next}$ com o primeiro vizinho alcançável de $S^{1}_{curr}$, são particularidades que visam explorar o potencial da Busca Tabu de gerar ótimas saídas sem requerer necessitar de sub-rotinas complexas.

Ainda, sobre as modificações propostas neste trabalho, cada uma das três variantes apresenta respectivamente o seguinte diferencial:

1. explora todas as vizinhanças por completo e não efetua a reconstrução parcial das rotas;
2. explora todas as vizinhanças por completo e efetua a reconstrução parcial das rotas de $S_{next}$ como forma de penalidade antes de iniciar a busca em uma vizinhança, substituindo assim a inicialização com o primeiro vizinho;
3. interrompe o ciclo de buscas se houver uma melhora de $S_{best}$ (avança para o próximo ciclo) e efetua o roteamento parcial (diversificação) estocasticamente, operação cuja probabilidade de acontecer é diretamente proporcional a $(K \bmod n)\,/K_{max}$.

Os valores de $\langle T, K_{max}\rangle$ foram definidos experimentalmente. Como não há um consenso sobre como obtê-los, optou-se por uma calibragem manual que se mostrasse atrativa para todo o conjunto de instâncias. Assim, chegou-se a três configurações: $\langle T, K_{max}\rangle = \langle n/3, 4n\rangle$ para uma busca mais restritiva, com possibilidade de avançar mais longe quanto estagnar em regiões de ótimo local, mas que pode descartar boas soluções com maior frequência; $\langle T, K_{max}\rangle = \langle n/6, 2n\rangle$ para uma busca mais dinâmica, permitindo aceitar mais soluções boas em relação à $S_{curr}$ porém levemente mais suscetível a ficar preso em torno de regiões de ótimo local; e $\langle T, K_{max}\rangle = \langle n^2/100, n\rangle$, para uma varredura rápida e pouco restritiva, visando direcionar a busca para regiões próximas que possam ter sido ignoradas. Os valores de $K_{max}$ foram definidos de forma que equilibrassem o tempo de execução, evitando *loops* demorados em torno da mesma solução.

### 3.5. Funções Auxiliares

Durantes os testes preliminares, ao comparar as soluções obtidas com as ótimas presentes nas instâncias de teste, notou-se que havia uma característica em comum: as soluções geradas eram sempre desbalanceadas, concentrando a maioria dos vértices nas primeiras rotas, enquanto que as soluções ótimas conhecidas eram melhor distribuídas.

Tal característica se deve à forma como a solução, o cálculo de custo total e a separação em rotas de acordo com o valor de $Q$ foram implementadas em código. Sendo os métodos aqui utilizados estritamente diretos – ou seja, sempre incluem o próximo vértice na rota atual se assim for possível, sem antes verificar posteriormente outras possibilidades –, o resultado foi um desbalanceamento intrínseco, não observado nas soluções documentadas que se visava atingir.

Como proposta de correção e melhoria dos resultados, foi desenvolvida uma função de balanceamento que percorre o vetor `S.route` no sentido inverso (do fim ao começo) e tenta mover os vértices de uma rota $i-1$ para a rota $i$ se o movimento for possível (restrição de capacidade) e isso representar uma melhoria no custo total. Para ilustração, considere a solução abaixo, oriunda da execução do algoritmo para a instância *A-n32-k5* de Augerat et al. [1998]:

```
Route #1: 22 29 14 7 1 12 16 26 30 18
Route #2: 24 27 20 5 25 10 8
Route #3: 13 21 31 19 17 3 23
Route #4: 6 2 28 4 11 9
Route #5: 15
cost 1268
```

Ao executar o balanceamento de carga, de custo computacional $O(n) = n$, obtém-se

```
Route #1: 22 29 14 7 1 12 16 26 30
Route #2: 18 24 27 20 5 25 10 8
Route #3: 13 21 31 19 17 3 23
Route #4: 6 2 28 4 11 9
Route #5: 15
cost 1239
```

cujo custo de roteamento representa uma diminuição de 2,3% em comparação com a solução anterior. Observe que apenas um movimento foi efetuado: o vértice 18 saiu do final da primeira rota para o início da segunda. Na codificação `S = {route, split, cost}` (ver seção 3.1), a única mudança seria no segundo índice do vetor `split`, que mudaria seu valor de 10 para 9.

Outra operação de baixo custo que potencializa a busca por soluções melhores foi a inversão completa da sequência de vértices em $S_0$. Considerando as decisões de implementação efetuadas, como a inicialização de $S_{next}$ com o primeiro vizinho e a interrupção do ciclo de buscas prematuramente, a inversão da rota permite que as buscas comecem por áreas opostas dentro da mesma vizinhança e avancem de formas distintas.

Denotando $S_0$ como a rota inicial gerada por $MetodoGuloso()$, $S_{dir}$ como a saída "direta" de $BuscaLocal(S_0)$, $S_{inv}$ como a saída de $BuscaLocal(Inverso(S_0))$, $S^*_{dir}$ e $S^*_{inv}$ como as melhores soluções obtidas em $BuscaTabu(S_{dir})$ e $BuscaTabu(S_{inv})$, concluiu-se que:

- a solução invertida $S_{inv}$ é melhor ou igual a $S_{dir}$ em cerca de metade dos casos de teste;
- a solução $S^*_{inv}$ obtida a partir da inversa é melhor ou igual a $S^*_{dir}$ em aproximadamente metade dos casos;
- nem sempre que $c_{total}(S_{dir}) < c_{total}(S_{inv})$ obtém-se $c_{total}(S^*_{dir}) < c_{total}(S^*_{inv})$.

Dada a impossibilidade de se estabelecer um padrão que permitisse optar por um tipo de rota e descartar o outro, ambas são consideradas na resolução do problema. Dessa forma, as três variações da BT descritas anteriormente são executadas para ambas as entradas – direta e invertida.

A terceira e última função auxiliar que foi agregada aos métodos de resolução difere-se um pouco das duas anteriores quanto ao resultado e ao propósito. Caracterizada por realizar movimentações intra-rotas, o *Roteamento Parcial* efetua uma reconstrução de cada rota através do *método do vizinho mais próximo* – semelhante à heurística de construção implementada –, mas sem verificar se o novo arranjo será melhor que o anterior.

De maneira intuitiva, dificilmente o resultado desse método, aplicado a uma solução oriunda de quaisquer das heurísticas ou metaheurísticas mencionadas nesse trabalho, será melhor que a solução de entrada. O grande destaque, portanto, não consiste em gerar uma solução melhor imediatamente, mas sim piorar uma solução atual e diversificar o espaço de busca, sem se afastar demais da solução anterior. De fato, ao inserir esse método como uma função de penalidade em $S_{next}$ de forma sistemática (na variação 2 da Busca Tabu) ou aleatória (na variação 3), foi possível alcançar os resultados ótimos que a variação 1 da metaheurística não alcançou. E, nos casos excepcionais, em que essa manipulação gera uma solução melhor que a anterior, a busca avança para uma região que não seria alcançada tão rapidamente pelas vizinhanças adotadas.

## 4. Experimentação e Análise dos Resultados

Os algoritmos descritos na seção 3 foram implementados na linguagem C++ e testados em um *hardware* com processador de 64 *bits* Intel® Core™ i5-7200U de 2,50 GHz, nas instâncias de Augerat et al. [1998] obtidas em *http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/*. Os arquivos são nomeados no formato $\gamma$-*n*$\alpha$-*k*$\beta$, onde $\gamma$ é o subconjunto de instâncias (A, B ou P), $\alpha$ é o número de vértices, incluindo o depósito, e $\beta$ é o número mínimo de veículos necessários para formar uma solução válida, obtido através da fórmula $\beta = \left\lceil \frac{1}{Q} \cdot \sum_{i=1}^{\alpha-1} q_i \right\rceil$.

Para cada implementação variante da Busca Tabu, foi experimentada a resolução das instâncias para as 3 configurações de parâmetros $\langle T, K \rangle$ e as duas opções de solução inicial $S_{dir}$ e $S_{inv}$, totalizando 6 execuções. Quanto à solução inicial que é repassada ao algoritmo, não foi possível determinar algum padrão que permitisse definir a melhor e, dessa forma, ambas (direta e inversa) foram mantidas como tentativas válidas. Em relação à $\langle T, K \rangle$, concluiu-se que aplicar a busca em cascata – usando a saída de uma Busca Tabu com um parâmetro para iniciar a tentativa com o próximo parâmetro – foi eficaz em maximizar a taxa de sucesso do método. Em outras palavras, para cada variante da metaheurística, foi feito uma varredura do mais restritivo ao mais flexível, seguindo o seguinte fluxo $S_0 \rightarrow BuscaTabu(\langle n/3, 4n \rangle) \rightarrow BuscaTabu(\langle n/6, 2n \rangle) \rightarrow BuscaTabu(\langle n^2/100, n \rangle) \rightarrow S_{final}$.

A tabela 1 contém os resultados obtidos nas três etapas de resolução para as instâncias do subconjunto A. As colunas de *Busca Tabu* referem-se às variações da metaheurística e a coluna *Tempo* representa o tempo gasto no processamento de cada metaheurística proposta. Para a variante III, foi estabelecido um número de repetições igual a 100, ou seja, a varredura com as 6 execuções foi repetida 100 vezes. É importante enfatizar que as variantes I e II são determinísticas, e por isso uma tentativa basta para determinar o melhor resultado alcançável.

Analisando a tabela 1, é possível extrair informações quanto ao desempenho específico de cada método implementado na resolução do PRVC. Em relação às heurísticas de construção e refinamento, tem-se o resultado esperado: soluções boas e válidas, mas razoavelmente distantes do ótimo conhecido. Para as implementações I e II da BT, os resultados são muito próximos daqueles conhecidos e tidos como objetivo a ser atingido, e apesar de não representar o melhor resultado dentre todos, o curto tempo de execução aliado com o fator determinístico caracterizam esses métodos

Tabela 1: Resultados obtidos para o subconjunto A de instâncias de Augerat et al. [1998]

| Instância | Ótimo | Método Guloso | Busca Local | Busca Tabu I | | Busca Tabu II | | Busca Tabu III | | Menor Erro |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | custo | tempo (s) | custo | tempo (s) | custo | tempo (s) | |
| A-n32-k5 | 784 | 1308 | 1221 | 786 | 3.14 | **784** | 2.36 | **784** | 46.19 | **0.00**% |
| A-n33-k5 | 661 | 1250 | 1127 | **661** | 3.61 | **661** | 5.43 | **661** | 54.10 | **0.00**% |
| A-n33-k6 | 742 | 1106 | 1055 | 743 | 3.46 | **742** | 3.22 | **742** | 46.85 | **0.00**% |
| A-n34-k5 | 778 | 1041 | 962 | 787 | 4.93 | 787 | 3.14 | **778** | 56.05 | **0.00**% |
| A-n36-k5 | 799 | 1109 | 1020 | 825 | 4.38 | **799** | 7.16 | **799** | 60.96 | **0.00**% |
| A-n37-k5 | 669 | 1047 | 997 | 687 | 5.15 | 698 | 5.61 | **669** | 78.57 | **0.00**% |
| A-n37-k6 | 949 | 1371 | 1280 | 963 | 5.80 | 975 | 4.50 | **949** | 86.92 | **0.00**% |
| A-n38-k5 | 730 | 1322 | 1232 | 735 | 5.70 | 734 | 7.94 | **730** | 89.04 | **0.00**% |
| A-n39-k5 | 822 | 1330 | 1262 | 831 | 9.41 | 825 | 7.07 | **822** | 91.23 | **0.00**% |
| A-n39-k6 | 831 | 1283 | 1189 | 833 | 6.48 | 834 | 7.70 | 833 | 90.61 | 0.24% |
| A-n44-k7 | 937 | 1564 | 1397 | 982 | 10.88 | 939 | 11.87 | **937** | 139.01 | **0.00**% |
| A-n45-k6 | 944 | 1624 | 1541 | 969 | 14.78 | 959 | 21.80 | 948 | 207.11 | 0.42% |
| A-n45-k7 | 1146 | 1489 | 1465 | 1165 | 15.22 | 1161 | 11.58 | **1146** | 192.31 | **0.00**% |
| A-n46-k7 | 914 | 1375 | 1308 | 974 | 14.46 | **914** | 21.45 | **914** | 215.04 | **0.00**% |
| A-n48-k7 | 1073 | 1585 | 1521 | 1117 | 21.44 | 1113 | 17.60 | **1073** | 214.60 | **0.00**% |
| A-n53-k7 | 1010 | 1541 | 1489 | 1054 | 30.94 | 1033 | 38.03 | **1010** | 358.20 | **0.00**% |
| A-n54-k7 | 1167 | 1583 | 1482 | 1189 | 36.02 | 1181 | 22.08 | **1167** | 379.14 | **0.00**% |
| A-n55-k9 | 1073 | 1557 | 1495 | 1111 | 37.78 | 1110 | 31.20 | 1074 | 488.30 | 0.09% |
| A-n60-k9 | 1408 | 2020 | 1950 | **1367** | 37.81 | **1360** | 44.68 | **1358** | 512.77 | **-3.55**% |
| A-n61-k9 | 1035 | 1710 | 1618 | 1036 | 60.09 | 1040 | 68.83 | **1035** | 757.95 | **0.00**% |
| A-n62-k8 | 1290 | 2004 | 1906 | 1320 | 68.21 | 1311 | 30.42 | 1308 | 602.22 | 1.40% |
| A-n63-k9 | 1634 | 2312 | 2265 | 1690 | 45.40 | 1646 | 43.69 | **1627** | 795.14 | **-0.43**% |
| A-n63-k10 | 1315 | 2178 | 2015 | 1404 | 46.82 | 1384 | 40.09 | 1325 | 858.95 | 0.76% |
| A-n64-k9 | 1402 | 2066 | 2023 | 1432 | 52.45 | 1441 | 53.11 | 1414 | 855.67 | 0.86% |
| A-n65-k9 | 1177 | 2030 | 1908 | 1199 | 85.12 | **1177** | 61.46 | 1181 | 821.23 | **0.00**% |
| A-n69-k9 | 1168 | 1771 | 1727 | 1191 | 68.23 | 1184 | 69.50 | **1163** | 851.63 | **-0.43**% |
| A-n80-k10 | 1764 | 2483 | 2386 | 1827 | 155.09 | 1828 | 176.97 | 1780 | 1716.03 | 0.91% |

como boas alternativas para a solução do problema, e são melhor recomendados para situações de tempo crítico ou baixa capacidade computacional. O destaque se mostra na terceira variante da metaheurística, que conseguiu alcançar o melhor resultado na maioria dos casos dentre as 74 instâncias de teste, e tem como único contraponto o fato de requerer uma grande quantidade de tentativas – foram utilizadas 100, mas quanto mais tentativas, maiores as chances de minimizar o erro –, o que se reflete em um aumento considerável do tempo de computação.

Outro ponto a ser analisado é referente à robustez do método. As três implementações variantes mantém uma qualidade alta nas soluções sem aumentar vertiginosamente o tempo de processamento conforme o aumento de $n$, construindo uma curva quase linear ao se relacionar essas grandezas. É possível notar melhor essa relação na tabela 2, em que os melhores resultados obtidos nas implementações propostas são comparados com os resultados de Kır et al. [2017], Awad e Elshaer [2018] e Harmanani et al. [2011]. Usou-se o tempo total para a solução proposta, já que está sendo considerado que os resultados são oriundos da execução conjunta das três variantes.

## 5. Conclusão

Ao longo desse trabalho foram vistas algumas abordagens de solução para o Problema de Roteamento de Veículos Capacitado. Algumas são inerentemente rápidas e de simples implementação, retornando configurações de rotas viáveis em tempo polinomial, embora costumem estar distantes da solução ótima. Outras, como é o caso das metaheurísticas, são mais abrangentes e têm a possibilidade de gerar soluções de alta qualidade.

Tabela 2: Comparação entre os resultados obtidos e os presentes na literatura

| Instância | Ótimo | Algoritmos de BT Propostos (melhor caso) | | Algoritmo Baseado em BT e ALNS de Kır et al. [2017] | | Algoritmo Genético de Awad e Elshaer [2018] | | *Simulated Annealing* de Harmanani et al. [2011] | |
|---|---|---|---|---|---|---|---|---|---|
| | | custo | tempo (s) | custo | tempo (s) | custo | tempo (s) | custo | tempo (s) |
| A-n32-k5 | 784 | 784 | 51.69 | 784 | 1.26 | 784 | – | 791.079 | – |
| A-n33-k5 | 661 | 661 | 63.14 | 661 | 1.32 | 661 | – | 685.577 | – |
| A-n33-k6 | 742 | 742 | 53.52 | 742 | 2.28 | 742 | – | 748.080 | – |
| A-n34-k5 | 778 | 778 | 64.12 | 778 | 2.98 | 778 | – | 802.614 | – |
| A-n36-k5 | 799 | 799 | 72.50 | 799 | 7.58 | 799 | – | 837.477 | – |
| A-n37-k5 | 669 | 669 | 89.33 | 669 | 12.65 | 669 | – | 697.943 | – |
| A-n37-k6 | 949 | 949 | 97.22 | 949 | 14.89 | 949 | – | 963.995 | – |
| A-n38-k5 | 730 | 730 | 102.67 | 730 | 22.32 | 730 | – | 733.945 | – |
| A-n39-k5 | 822 | 822 | 107.71 | 822 | 48.50 | 822 | – | 848.322 | – |
| A-n39-k6 | 831 | 833 | 104.79 | 831 | 44.62 | 831 | – | 859.147 | – |
| A-n44-k7 | 937 | 937 | 161.76 | 939 | 102.03 | 937 | – | 951.802 | – |
| A-n45-k6 | 944 | 948 | 243.69 | 955 | 97.88 | 944 | – | 982.829 | – |
| A-n45-k7 | 1146 | 1146 | 219.10 | 1153 | 230.56 | 1148 | – | 1199.50 | – |
| A-n46-k7 | 914 | 914 | 250.95 | 915 | 201.93 | 914 | – | – | – |
| A-n48-k7 | 1073 | 1073 | 253.64 | 1073 | 298.76 | 1073 | – | – | – |
| A-n53-k7 | 1010 | 1010 | 427.16 | 1026 | 1987.48 | 1010 | – | 1044.91 | – |
| A-n54-k7 | 1167 | 1167 | 437.24 | 1169 | 2369.52 | 1172 | – | 1192.69 | – |
| A-n55-k9 | 1073 | 1074 | 557.28 | 1074 | 3892.10 | 1073 | – | 1144.85 | – |
| A-n60-k9 | 1408 | 1358 | 595.25 | 1366 | 7760.19 | 1354 | – | 1455.63 | – |
| A-n61-k9 | 1035 | 1035 | 886.86 | 1045 | 7239.88 | 1037 | – | 1064.76 | – |
| A-n62-k8 | 1290 | 1308 | 700.85 | 1302 | 12331.62 | 1299 | – | 1484.81 | – |
| A-n63-k9 | 1634 | 1627 | 884.23 | 1644 | 11999.34 | 1621 | – | 1698.63 | – |
| A-n63-k10 | 1315 | 1325 | 945.86 | 1325 | 11906.02 | 1319 | – | – | – |
| A-n64-k9 | 1402 | 1414 | 961.23 | 1442 | 17002.33 | 1418 | – | – | – |
| A-n65-k9 | 1177 | 1177 | 967.80 | 1189 | 18071.70 | 1177 | – | – | – |
| A-n69-k9 | 1168 | 1163 | 989.35 | 1169 | 18720.03 | 1169 | – | – | – |
| A-n80-k10 | 1764 | 1780 | 2048.08 | 1790 | 20692.47 | 1804 | – | – | – |

As extensões propostas e implementadas para a BT atingiram resultados ótimos para 51 das 74 instâncias testadas, e mesmo para aquelas que não se atingiu o objetivo, os maiores erros percentuais foram de 1,40%, 1,36% e 10,31% – para os subconjuntos A, B e P, respectivamente –, com um erro médio de 0,8% (análise feita considerando o melhor resultado obtido dentre os 3 métodos). Conforme visto na tabela 1, a maioria dos melhores resultados são oriundos da variante 3 da BT, mas mesmo os resultados não ótimos obtidos nas demais implementações possuem um baixo erro percentual, e considerando o tempo baixo de processamento que foi utilizado esses métodos representam alternativas viáveis para situações de tempo crítico ou menor capacidade computacional.

Há ainda outros recursos a serem testados, como as estruturas de memória de longo prazo e estratégias adaptativas Glover et al. [1993]; Gendreau [2003], mas por mais rebuscada que possa ser a técnica adotada, a parte crucial e que demanda mais atenção permanece sendo a escolha das vizinhanças e o ajuste de parâmetros. No geral, o que se pode concluir é aquilo que diversos autores reiteram: a Busca Tabu é uma metaheurística poderosa e eficiente na solução do PRVC.

## Referências

Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., e Rinaldi, G. (1998). Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report R.495, IASI-CNR.

Awad, H. e Elshaer, R. (2018). An effective genetic algorithm for capacitated vehicle routing problem.

Cunha, C. B. d., Bonasser, U. d. O., e Abrahão, F. T. M. (2002). Experimentos computacionais com heurísticas de melhorias para o problema do caixeiro viajante. In *Congresso de Pesquisa e Ensino em Transportes*. ANPET.

Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1): 80–91. URL https://doi.org/10.1287/mnsc.6.1.80.

Faulin, J., Juan, A., Lera-Lopez, F., e Grasman, S. (2011). Solving the capacitated vehicle routing problem with environmental criteria based on real estimations in road transportation: A case study. *Procedia - Social and Behavioral Sciences*, 20:323–334.

Gendreau, M. (2003). *An Introduction to Tabu Search*, p. 37–54. Springer US, Boston, MA. ISBN 978-0-306-48056-0. URL https://doi.org/10.1007/0-306-48056-5_2.

Gendreau, M. e Potvin, J.-Y. (2010). *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition. ISBN 1441916636.

Glover, F., Taillard, E., e Taillard, E. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41(1):1–28. ISSN 1572-9338. URL https://doi.org/10.1007/BF02078647.

Golden, B., Raghavan, S., e Wasil, E., editors (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, USA. ISBN 9780387777771.

Harmanani, H., Azar, D., Helal, N., e Keirouz, W. (2011). A simulated annealing algorithm for the capacitated vehicle routing problem. p. 96–101.

Ibrahim, A., Abdulaziz, R., e Ishaya, J. (2019). Capacitated vehicle routing problem. *International Journal of Research - GRANTHAALAYAH*, 7:310 – 327.

Kır, S., Yazgan, H., e Tüncel, E. (2017). A novel heuristic algorithm for capacitated vehicle routing problem. *Journal of Industrial Engineering International*, 13.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345 – 358. ISSN 0377-2217.

Papadimitriou, C. H. e Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., USA. ISBN 0131524623.

Ralphs, T. K., Kopman, L., Pulleyblank, W. R., e Trotter, L. E. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, 94:343–359.

Simas, E. P. L. (2007). Utilizando a busca tabu na resolução do problema de roteamento de veículos. Master's thesis, Universidade do Vale do Rio do Sinos.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing. ISBN 0470278587.

Toth, P. e Vigo, D., editors (2001). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, USA. ISBN 0898714982.

Galoá { Este trabalho foi publicado utilizando o **Galoá** proceedings