A Variable Neighborhood Search Heuristic for the Traveling Salesman Problem with Hotel Selection

Marques M. Sousa, Luiz Satoru Ochi
Instituto de Computação
Universidade Federal Fluminense
Niterói, RJ, Brasil
Email: {msousa, satoru}@ic.uff.br

Igor Machado Coelho
Departamento de Computação
Universidade do Estado do Rio de Janeiro
Rio de Janeiro, RJ, Brasil
Email: igor.machado@ime.uerj.br

Luciana Brugiolo Gonçalves
Dep. Ciência da Computação
Universidade Federal de Juiz de Fora
Juiz de Fora, MG, Brasil
Email: lbrugiolo@ice.ufjf.br

***Abstract*—This work deals with the Traveling Salesman Problem with Hotel Selection (TSPHS), a variant of the classic Traveling Salesman Problem (TSP). In the TSPHS, a set of hotels can be visited in strategic points of the route, dividing it in a minimum number of trips. Each trip must not exceed a given time limit, minimizing also the total time traveled. The TSPHS is $\mathcal{NP}$-Hard, being a generalization of the TSP, so the main approaches in literature are based in Mathematical Programming and Metaheuristics. The metaheuristics are generic heuristics capable of escaping from local optima, usually obtaining good quality solutions in low computational time. It is developed a heuristic based on Variable Neighborhood Search, compared with the best algorithms in literature using classic instances. Computational results indicate that the proposed algorithm finds solutions with fewer trips in low computational time, with a traveled total time comparable to the best known solutions.**

***Keywords*—Traveling Salesman Problem with Hotel Selection, Variable Neighborhood Search, Metaheuristics.**

I. Introdução

O clássico Problema do Caixeiro Viajante (PCV) é um dos problemas mais estudados no campo da otimização combinatória [1]. A possibilidade de aplicação deste problema no mundo real, fez com que diversas variantes fossem propostas [2]. Neste trabalho, é abordada uma variante do PCV denominada Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH), que além de considerar as características originais do problema no qual foi derivado, ainda lida com a inserção de hotéis ao longo da rota percorrida pelo caixeiro de forma a não extrapolar o limite de tempo diário máximo a ser percorrido.

Ao longo deste trabalho serão utilizados os termos **rota** e **viagem**. Uma viagem representa o atendimento a um subconjunto de clientes em uma dada jornada de trabalho que inicia e termina em algum dos hotéis disponíveis, considerando um limite de tempo máximo previamente determinado. O termo rota abrange o conjunto de todas as viagens que foram necessárias para atender à todos os clientes.

No PCV, o objetivo consiste em determinar um ciclo hamiltoniano de custo mínimo, porém todos os clientes são atendidos sem levar em consideração um limite de tempo para a jornada de trabalho do caixeiro. Já no PCVSH, há situações em que o limite de tempo para a realização de uma jornada de trabalho pode não ser suficiente para realizar a visita a todos os clientes. Neste caso, torna-se necessário, ao final de uma jornada, procurar um hotel para aguardar o início de uma nova jornada, de onde será possível continuar o atendimento aos clientes no dia seguinte. No final de cada jornada, deve ser tomada a decisão de qual hotel deve ser escolhido para realizar a parada.

Este problema foi proposto em 2012 por Vansteenwegen et al. [3] e, assim como o PCV, também é da classe $\mathcal{NP}$-Difícil. Esta similaridade pode ser comprovada de forma simples, ou seja, bastando utilizar o limite de tempo grande o suficiente para as viagens fazendo com que todos os clientes possam ser atendidos com uma única viagem. Desta maneira, nenhum hotel intermediário é utilizado durante a rota e o PCVSH se resume ao PCV. O objetivo do problema é encontrar a melhor ordem de clientes usando os hotéis necessários de forma que a rota seja particionada em viagens viáveis que atendem ao limite de tempo pré-definido, minimizando o tempo total necessário para visitar todos os clientes e o número de viagens necessárias.

Técnicas de programação matemática são empregadas na literatura [3], [4]. Porém, tais abordagens são limitadas a problemas-teste de pequeno porte, nos quais apenas um pequeno conjunto de clientes e hotéis são considerados. Estratégias baseadas em metaheurísticas tem se mostrado promissoras para lidar com problemas maiores com um baixo tempo computacional. Em especial, a metaheurística *Variable Neighborhood Search* (VNS) tem se mostrado poderosa na resolução de problemas de roteamento de grande porte [5].

Neste trabalho, propõe-se o desenvolvimento de uma nova heurística inspirada em VNS para a resolução do PCVSH, denominada *Client Perturbation Variable Neighborhood Search* (CP-VNS). Uma comparação de resultados é feita com os melhores e mais rápidos algoritmos da literatura para o PCVSH, incluindo o Algoritmo Memético de Castro et al. [4]. São considerados problemas-teste da literatura classificados de pequeno, médio a grande porte, contendo de 10 a 1002 clientes.

O restante do artigo é organizado em cinco seções. A Seção 2 fornece uma breve descrição dos trabalhos relacionados. Na Seção 3 é apresentada uma formulação matemática para o problema. A Seção 4 descreve a abordagem heurística proposta CP-VNS, inspirada pela metaheurística *Variable Neighborhood Search*. Finalmente, a Seção 5 detalha os experimentos computacionais realizados e a Seção 6 apresenta as conclusões sobre o presente trabalho.

978-1-4673-9143-6/15/$31.00

II. Trabalhos Relacionados

O Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) foi proposto por Vansteenwegen et al. [3], e desde então diversos trabalhos têm sido desenvolvidos sobre o assunto. Os autores apresentaram um modelo matemático de Programação Linear Inteira Mista. Ainda destacaram a diferença entre o PCVSH e outros problemas de otimização da literatura. Além do modelo, foi apresentada uma heurística que utiliza dois procedimentos para construção da solução inicial e um procedimento de melhoria composto por diferentes operadores de busca local. Por fim, um conjunto de problemas-teste com diferentes tamanhos e níveis de dificuldade foi introduzido. Neste primeiro trabalho, os autores compararam os resultados obtidos pela heurística com o resultado obtido pelo modelo matemático.

Dando continuidade à pesquisa sobre o PCVSH, os autores em [4] pela primeira vez tratam o problema utilizando uma metaheurística. A metaheurística desenvolvida foi uma combinação da metaheurística *Greedy Randomized Adaptive Search Procedure* (GRASP), proposta em 1995 por Feo e Resende [6], com um procedimento *Variable Neighbourhood Descent* (VND) como etapa de busca local. O VND utilizado consiste na aplicação de uma série de estruturas de vizinhança, cujo resultado é uma solução localmente ótima em relação às estruturas de vizinhança consideradas [7]. Para o conjunto de problemas-teste contido na literatura, a heurística provou ser capaz de encontrar, quase na totalidade, os melhores resultados conhecidos até o momento e ainda conseguiu encontrar melhores soluções para alguns problemas-teste.

Partindo para um tipo de estratégia ainda não testada para tratar o problema, foi utilizada uma metaheurística com característica populacional [4]. Nesta abordagem, denominada Algoritmo Memético (MA), as operações básicas de um Algoritmo Genético como cruzamento e mutação dos indivíduos da população produzem indivíduos que são aprimorados por estratégias de busca local [8]. Castro et al. [4] utilizaram apenas operações que alteram os hotéis no MA. Para a definição das melhores posições para cada cliente foi utilizada uma heurística baseada na Busca Tabu [9], [10]. Os autores utilizaram estruturas de vizinhança e heurísticas de geração da solução inicial conhecidas na literatura com bons resultados em trabalhos anteriores [3], [11]. As soluções obtidas superaram as encontradas em trabalhos anteriores que abordaram o PCVSH.

Também com uma estratégia baseada em Algoritmo Memético, Sousa e Gonçalves [12] propõem dois Algoritmos Meméticos, sendo um com Busca Tabu (BT) e outro com um VND de ordem aleatória (RVND). As estratégias para definição das soluções iniciais e de exploração das vizinhanças foram as mesmas utilizadas nos trabalhos anteriores. Os resultados foram comparados com os obtidos anteriormente [4] e foram encontrados para alguns problemas-teste resultados melhores utilizando a estratégia com BT. Com o RVND, para a maioria dos problemas-teste, o tempo computacional foi reduzido e a qualidade da solução foi semelhante ou até melhor que as até então conhecidas.

Com o intuito de se encontrar soluções para os problemas-teste em um tempo computacional baixo, foi desenvolvida uma heurística baseada em *Iterated Local Search* (ILS) capaz de alcançar, em tempos computacionais bem reduzidos, boas soluções para todos os problemas-teste [13]. Os autores propuseram uma nova formulação matemática para o PCVSH baseada na utilização de viagens, ao invés de arestas. A criação de uma solução inicial pela heurística proposta utiliza um método *order-first split-second*, que em um primeiro momento, define uma solução para o PCV e, posteriormente, considerando as restrições de tempo particiona esta solução tornando-a uma solução válida para o PCVSH. Este método de geração da solução inicial é combinado com uma heurística *Iterated Local Search* (ILS), que consiste em uma fase de perturbação seguida pela busca local [14]. A etapa de busca local foi substituída pelo procedimento VND, capaz de escapar de ótimos locais por meio de uma troca sistemática de estruturas de vizinhança [15]. Como estrutura de perturbação do ILS, foram usadas duas estratégias. A primeira, aplica mudanças apenas nos clientes, e a segunda modifica apenas os hotéis intermediários da solução. As estruturas de vizinhança utilizadas no VND foram as mesmas utilizadas em trabalhos anteriores. A proposta encontrou resultados competitivos quanto a qualidade da solução e também um tempo computacional inferior ao encontrado em todos os trabalhos conhecidos até então.

Como variante do PCVSH, pode-se citar o Problema do Caixeiro Viajante com Múltiplas Janelas de Tempo e Seleção de Hotéis (PCV-MJTSH), variante do PCVSH com características particulares [16]. O objetivo nesta variante é minimizar os custos da rota, que envolvem o salário do caixeiro, as despesas com hotéis, despesas de viagem e as taxas de penalização para os clientes que não forem atendidos. Os autores apresentaram um modelo de programação linear inteira mista e uma abordagem heurística que combina estratégias de inserção mais barata, 2-opt e reinício randomizado. Para pequenas instâncias o modelo encontrou as soluções ótimas em tempo razoável. A heurística por sua vez, obtém as soluções encontradas pelo modelo matemático nos problemas-teste pequenos e provou ser rápida, eficiente e eficaz para grandes instâncias.

Outro problema que possui grande similaridade com o PCVSH é o Problema do Caixeiro Viajante com Grupamentos (PCVG), que considera *clusters* de clientes espalhados geograficamente [17]. O PCVG possui como objetivo encontrar um ciclo hamiltoniano de custo minímo que passe por todos os vértices do grafo associado e visite os vértices de cada grupo de forma contígua. Esta variante do PCV apresenta grande similaridade uma vez que pode-se assimilar a construção das viagens no PCVSH com a definição dos grupos no PCVG. A escolha da ligação entre dois grupos quaisquer é equivalente à escolha do melhor hotel entre duas viagens no PCVSH. Diversos autores realizaram trabalhos acerca do PCVG desde seu surgimento [18]. Nesse trabalho, o autor utiliza metaheurísticas como GRASP e ILS, além de, produzir um algoritmo exato para tratar tal problema. Ambas as abordagens propostas apresentaram bons resultados quando comparadas às existentes na literatura.

O PCVSH também se relaciona com o Problema de Roteamento de Veículos (PRV), no qual um conjunto de clientes deve ser atendido por diversas viagens partindo de um depósito comum [19]. Neste caso, o PCVSH pode ser visto como um PRV multi-nível, em que cada ponto de parada (hotel) é determinado *a-priori*, e em um segundo momento o roteamento é efetuado para toda a rota. Desta forma, o PCVSH pode

ser aplicado, na prática, em problemas relacionados a entrega de mercadorias e transporte de cargas. Quando o número de viagens é fixo e não é possível visitar todos os clientes, ele se torna um problema de *orienteering*. Sendo que neste caso, a função objetivo prioriza clientes com um maior benefício associado [20].

III. Formulação Matemática

Para definição formal do Problema do Caixeiro Viajante com Seleção Hotéis (PCVSH), considere um grafo $G = (V, A)$ onde $V = H \cup C$, sendo H o conjunto não vazio que representa os hotéis e C o conjunto que representa os clientes que devem ser visitados. No conjunto $A = \{(i,j)|i,j \in V, i \neq j\}$, cada aresta (i,j) representa a ligação entre os clientes/hotéis i e j. Cada cliente $i \in C$ requer um tempo de serviço ou tempo de visita τ_i (com $\tau_i = 0$, para todo $i \in H$). O tempo c_{ij} necessário para viajar da localidade i para j é conhecido para todos os pares de localidades. O mesmo hotel $(i = 0, i \in H)$ deve iniciar e finalizar a rota, podendo ser utilizado como hotel intermediário entre viagens. Os outros hotéis contidos em H, podem ser utilizados quando forem necessários para constituir a rota. Como um hotel H pode ser visitado várias vezes na mesma rota, a solução do PCVSH pode ou não ser representada por um ciclo simples.

A Figura 1 apresenta um exemplo de rota válida para um PCVSH. O caixeiro parte do hotel inicial (quadrado) e visita alguns clientes (círculos). Uma parada é feita em um hotel (losango), de onde deve continuar a próxima rota. A visita aos hotéis não é obrigatória, sendo possível repetir hotéis ao longo da rota, como também terminar sem necessariamente utilizar todos os hotéis disponíveis.

Como restrições adicionais, tem-se que cada viagem deve iniciar e finalizar em um dos hotéis disponíveis, o tempo total percorrido em uma viagem não pode exceder o tempo limite L e por fim, uma viagem deve iniciar em um hotel onde a viagem prévia terminou.

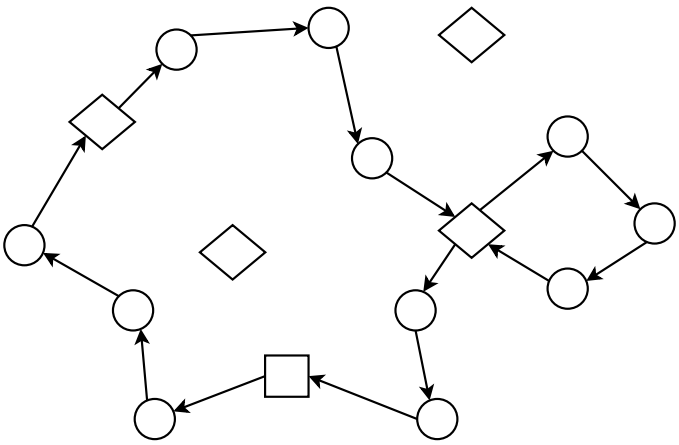

Figura 1. Rota para um PCVSH: o quadrado representa o hotel inicial e final, os círculos representam clientes e os losangos representam os hotéis.

Neste trabalho foi proposta uma alteração no modelo de Programação Linear Inteira (PLI) proposto anteriormente [4], com o intuito de fornecer uma formulação alternativa. Diferentemente da formulação original com eliminação de sub-ciclos por subconjuntos, que necessita de um número exponencial de restrições, nesta proposta é utilizado um modelo baseado em fluxo, com um número polinomial de variáveis e restrições. O fluxo de entrada nos vértices intermediários deve ser sempre menor que o fluxo de saída desses vértices, dessa forma nenhum subciclo pode ser formado. Para o hotel inicial e final, foram criadas restrições específicas tratando o fluxo. Com o intuito de priorizar rotas que contenham um número menor de viagens, uma constante M suficientemente grande deve ser utilizada na função objetivo.

Seja x_{ij}^d uma variável binária que recebe o valor 1 se, na d-ésima viagem, uma visita a um cliente/hotel i é seguida pela visita a um cliente/hotel j ou o valor 0, caso contrário. A variável binária y^d receberá o valor 1 se na viagem d no mínimo um cliente/hotel for visitado ou 0, caso contrário. Assim, y^d receberá o valor zero se nenhuma viagem for necessária no dia d. As variáveis y^d e x_{ij}^d são usadas na função objetivo, respectivamente, para minimizar o número de viagens e o tempo total percorrido. Por fim, a constante D representando o número máximo de viagens possíveis em uma solução foi definida pela solução heurística contida na literatura [4]. Uma variável f_{ij} representa o valor do fluxo que passará pela aresta que liga os vértices i e j.

$$\min \quad M \sum_{d=1}^{D} y^d + \sum_{d=1}^{D} \sum_{(i,j)\in A} c_{ij} x_{ij}^d \tag{1}$$

Sujeito a:

$$\sum_{d=1}^{D} \sum_{i \in V} x_{ij}^d = 1, \forall j \in C \tag{2}$$

$$\sum_{i \in V} x_{ij}^d = \sum_{i \in V} x_{ji}^d, \forall j \in C, \forall d = 1, ..., D \tag{3}$$

$$\sum_{h \in H} \sum_{j \in V \setminus \{h\}} x_{hj}^d = y^d, \forall d = 1, ..., D \tag{4}$$

$$\sum_{h \in H} \sum_{i \in V \setminus \{h\}} x_{ih}^d = y^d, \forall d = 1, ..., D \tag{5}$$

$$\sum_{(i,j)\in A} (c_{ij} + \tau_j) x_{ij}^d \leq L, \forall d = 1, ..., D \tag{6}$$

$$\sum_{j \in V \setminus \{0\}} x_{0j}^1 = 1 \tag{7}$$

$$\sum_{i \in V \setminus \{0\}} x_{i0}^d \geq y^d - y^{d+1}, \forall d = 1, ..., D-1 \tag{8}$$

$$\sum_{i \in V} x_{ih}^d + y^d \geq \sum_{i \in V} x_{hi}^{d+1} + y^{d+1}, \quad \forall h \in H, \forall d = 1, ..., D-1 \tag{9}$$

$$\sum_{i \in V} x_{ih}^d - \sum_{i \in V} x_{hi}^{d+1} \leq 1 - y^{d+1}, \quad \forall h \in H, \forall d = 1, ..., D-1 \tag{10}$$

$$x_{ij}^{d} \leq y^{d}, \forall (i,j) \in A, \forall d = 1, ..., D \quad (11)$$

$$y^{d} \geq y^{d+1}, \forall d = 1, ..., D-1 \quad (12)$$

$$f_{0j} = x_{0j}^{1}, \forall j \in V \backslash \{0\} \quad (13)$$

$$\sum_{j \in V} f_{ij} = \sum_{j \in V} f_{ji} + \sum_{d=1}^{D} \sum_{j \in V} x_{ji}^{d}, \forall i \in V \backslash \{0\} \quad (14)$$

$$f_{ij} \leq (|C| + |D|) \sum_{d=1}^{D} x_{ij}^{d}, \forall (i,j) \in V \quad (15)$$

$$f_{ij} \geq 0, (i,j) \in A | i \neq j \quad (16)$$

$$x_{ij}^{d} \in \{0,1\}, \forall (i,j) \in A, \forall d = 1, ..., D \quad (17)$$

$$y^{d} \in \{0,1\}, \forall d = 1, ..., D \quad (18)$$

No modelo matemático, a função objetivo (1) visa minimizar o número de viagens e o tempo total percorrido. As restrições (2) garantem que cada cliente será visitado exatamente uma vez. As restrições (3) garantem que haja conectividade entre cada viagem contida na rota e as restrições (4) e (5) garantem que cada viagem inicia e termina em um dos H hotéis disponíveis. As restrições (6) impõem um limite ao tempo total de uma viagem, incluindo tempo gasto para percorrer o caminho e os tempos de visita aos clientes. As restrições (7) e (8) definem que a rota deve iniciar e terminar no hotel 0. As restrições (9) e (10) indicam que se uma viagem termina em um dado hotel, então a próxima viagem deve, obrigatoriamente, iniciar neste hotel. As restrições (11) definem que uma viagem está sendo utilizada somente se acontece pelo menos uma visita a um cliente ou a um hotel naquela jornada de trabalho. Restrições (12) garantem que as viagens serão realizadas em dias consecutivos, iniciando no primeiro dia. As restrições (13) definem que o fluxo na aresta que sai do hotel 0 para qualquer outro vértice na primeira viagem será igual a 1, garantindo assim, que saia apenas uma aresta do hotel inicial na primeira jornada de trabalho. As restrições (14) garantem que o fluxo de entrada em um vértice i será sempre uma unidade menor que o fluxo de saída. As restrições (15) garantem que se uma determinada aresta entre i e j não for utilizada, então o valor do fluxo naquela aresta será 0. Por fim, as restrições (16), (17) e (18) indicam o domínio das variáveis.

IV. *Client Perturbation Variable Neighborhood Search*

O modelo de programação matemática proposto é bastante interessante por sua capacidade de resolver o problema de forma exata com um número polinomial de variáveis e restrições, mas na prática este modelo só é capaz de provar a otimalidade em problemas-teste de pequeno porte. Com o intuito de tratar instâncias maiores do problema, uma estratégia de solução com aplicação de uma metaheurística é apresentada nesta seção. Esta estratégia combina a utilização da heurística de Lin-Kernighan [21] para geração de uma solução inicial viável para o problema, com a metaheurística *Variable Neighborhood Search* (VNS). O VNS é capaz de escapar de ótimos locais por meio de uma troca sistemática das estruturas de vizinhança exploradas [22].

A estrutura da metaheurística proposta *Client Perturbation Variable Neigborhood Search* (CP-VNS) é apresentada no Algoritmo 1. Inicialmente, uma solução viável para o Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) que satisfaz o tempo total limite de todas as viagens contidas na rota é construída. Esta solução é gerada pela heurística de Lin-Kernighan, juntamente com um procedimento baseado no Algoritmo de Dijkstra [23]. Antes de iniciar o procedimento de aplicação do VNS, o procedimento *Variable Neighborhood Descent* (ver Subseção IV-D) é aplicado com o objetivo de refinar ainda mais a solução inicial. Com uma solução razoavelmente boa, o VNS é aplicado explorando sistematicamente as mudanças em uma dada vizinhança, buscando desta forma encontrar soluções ótimas locais e garantindo que haja uma estratégia eficiente de fuga quando o algoritmo estiver preso nos ótimos locais.

O CP-VNS finaliza sua execução se durante i_{max} iterações não houver nenhuma melhora significativa na melhor solução corrente. A cada iteração, o algoritmo move a solução corrente para uma solução vizinha, aplicando um procedimento que realoca um dos hotéis intermediários para uma posição escolhida aleatoriamente. Para complementar a eficácia da exploração de soluções vizinhas (método *Vizinhança*), um outro procedimento é utilizado a cada determinado número de iterações (método *Perturbação*). Este procedimento efetua uma perturbação de um percentual θ_1 de clientes, buscando prover uma garantia maior de que seja possível escapar dos ótimos locais por meio da exploração de soluções em vizinhanças distintas da atual. Se em determinada iteração a metaheurística VND for capaz de melhorar a solução corrente, o contador de iterações é zerado. Caso contrário, é incrementado.

Algoritmo 1: CP-VNS(i_{max}, θ_1)

```
1  início
2      y ← LKH-Dijkstra()
3      VND(y)
4      i ← 0
5      enquanto (i < i_max) faça
6          se ((i % 5) ≠ 0) então
7              y' ← Vizinhança(y)
8          senão
9              y' ← Perturbação(y, θ_1)
10         fim
11         y' ← VND(y')
12         se y' melhor que y então
13             y ← y'
14             i ← 0
15         senão
16             i ← i + 1
17         fim
18     fim
19     Retornar y';
20 fim
```

A. *Construção da Solução Inicial*

A estratégia utilizada para construção da solução inicial é considerada como pertencente ao grupo de técnicas para geração de soluções *order-first split-second* que primeiramente ordenam os clientes em uma rota e posteriormente realizam a divisão introduzindo hotéis conforme necessário [24]. Inicialmente é aplicada a heurística de Lin-Kernighan [21] e logo em seguinda o Algoritmo de Dijkstra [23].

O primeiro passo para a criação da solução é construir uma rota de boa qualidade para o Problema do Caixeiro Viajante (PCV). Esta rota possui como hotéis inicial e final o hotel $H0$, além de, visitar todos os clientes sem considerar a restrição do limite de tempo total das viagens. Para construir a rota é utilizada a heurística de Lin-Kernighan da forma como foi implementada no solver Concorde [25].

A heurística de Lin-Kernighan é considerada como uma das melhores estratégias conhecidas para gerar soluções de boa qualidade. Entretanto, sua eficiência é proporcional ao trabalho necessário para realizar sua implementação, havendo várias maneiras diferentes de fazê-la. Logo, detalhes de sua implementação tem uma influência significativa no seu desempenho [26].

A rota gerada pela LKH, na maioria das vezes, não é viável para o PCVSH, logo é necessário aplicar um procedimento que particione a rota em viagens viáveis. A divisão da rota é realizada por meio do processo de divisão apresentado por Prins *et al.* [27], que baseia-se no Algoritmo de Dijkstra. O objetivo do método de divisão é particionar a rota por meio da inserção de hotéis intermediários, garantindo a viabilidade de cada viagem gerada.

Assim, considerando os clientes ordenados de acordo com as regras estabelecidas pela heurística LKH, no procedimento de divisão é construído um grafo auxiliar G' contendo $mn+1$ vértices, com $m = |H|$ e $n = |C|$. O primeiro nó do grafo representa o hotel inicial e é definido com o índice 1. O restante dos vértices do grafo representam uma combinação de hotéis e clientes. Para representar a visita a um hotel p que foi precedido pela visita ao i-ésimo cliente s_i, é utilizada a notação i^p. O nó final do grafo é definido como $V^{(mn+1)-(m-1)}$, e determina o hotel final da rota, depois que a visitação a todos os clientes foi realizada. Uma aresta (i^p, j^q) é adicionada ao grafo se uma viagem partindo do hotel p, visitando os clientes de s_{i+1} até s_j e chegando ao hotel q, for viável. Por fim, o peso da aresta (i^p, j^q) é caracterizado pela soma das distâncias necessárias para o caixeiro viajar de i^p para j^q, passando pelos clientes de s_{i+1} até s_j, somados os tempos de visita de cada cliente. Para cada aresta inserida no grafo, foi acrescido ao seu peso o valor de uma constante (γ) que deve ser um valor relativamente grande, definido de acordo com as instâncias a serem testadas e que contribuirá no momento de minimização do número de viagens. O peso destas arestas é expresso pela Equação (19).

O resultado retornado pela equação é composto pelo tempo necessário para percorrer o trajeto entre o hotel p e o cliente s_{i+1}, somando-se o tempo da viagem entre os clientes s_v e s_{v+1}, mais o tempo de visita associado a cada cliente s_v. Neste caso, a variável v assume os valores de $(i+1)$ à $(j-1)$. Por fim, é acrescentado o tempo de visita do cliente s_j, o tempo necessário para o caixeiro se deslocar do cliente s_j até o hotel q e o valor da constante γ. A soma total representa o peso associado a aresta (i^p, j^q).

$$c_{ps_{i+1}} + \sum_{v=i+1}^{j-1} (c_{s_v s_{v+1}} + \tau_{s_v}) + \tau_{s_j} + c_{s_j q} + \gamma \quad (19)$$

Após a construção do grafo auxiliar, o Algoritmo de Dijkstra é utilizado para encontrar o caminho mínimo entre os hotéis inicial e final. Como foi inserido no peso de cada aresta a constante γ, o caminho definido pelo Algoritmo de Dijkstra prioriza a utilização do menor número de viagens necessárias para atender a todos os clientes. Desta forma, é realizado simultaneamente a minimização do número de arestas utilizadas, que correspondem ao número de viagens e o tempo total da rota.

B. Perturbação da Solução Alterando os Clientes

O método de perturbação utilizado foi proposto por Castro *et al.* [13] e consiste na aplicação de passos simples que guiam a solução para uma nova vizinhança. Este método consiste na perturbação da solução considerando a mudança apenas dos clientes. A perturbação consiste na alteração da posição dos clientes na rota. O primeiro passo para sua aplicação é a definição de um parâmetro teta (θ_1) que indica quantos clientes da solução devem ter sua posição alterada na rota. O próximo passo é escolher de forma aleatória quais clientes terão suas posições alteradas e quais serão suas novas posições.

C. Estruturas de Exploração de Vizinhanças

As estruturas de vizinhança são as principais responsáveis pela melhora da solução ao longo das iterações do CP-VNS. Logo, foram utilizadas estruturas que trabalham otimizando a solução a nível de viagens (*intra-trips*), como também a nível da rota (*inter-trips*). As estruturas de vizinhança utilizadas são extensivamente utilizadas em problemas similares e impactam positivamente na melhora das soluções para o PCVSH. São elas:

- Estrutura *Relocate* (*inter-trips*) proposta por Laporte et al. [28], que realoca uma sequência de c clientes de uma viagem para outra distinta, sendo $c = 3, 2, 1$.
- Estrutura *Exchange* (*inter-trips*) proposta por Laporte et al. [28], que realiza a troca de uma sequência de c clientes entre duas viagens, sendo $c = 3, 2, 1$.
- Estrutura *Or-Opt* (*intra-trips*) proposta por Or [29], que realoca uma sequência de c clientes internamente a uma viagem, sendo $c = 3, 2, 1$.
- Estrutura *2-Opt* (*intra-trips*), que desconecta uma viagem em dois locais, reconectando-os de forma que a sequência de clientes interna as duas quebras seja invertida [30].
- Operador de hotéis *Join-trips* proposto por Castro et al. [4], que realiza a remoção de hotéis intermediários, ou seja, se é possível retirar um hotel de forma que a soma do tempo de sua viagem predecessora mais o tempo de sua sucessora não excedam o tempo limite L definido para uma viagem.
- Operador de hotéis *Change-hotels* proposto por Castro et al. [13], que testa sequencialmente a troca de cada um dos hotéis intermediários pelos outros hotéis disponíveis. Caso a troca resulte em uma melhora no tempo total da solução e não a torne inviável, a troca é efetuada.

A Figura 2 traz uma representação gráfica das estruturas de vizinhança utilizadas, onde as linhas tracejadas representam

as modificações que foram realizadas na rota do PCVSH. As representações seguem a ordem de definição das estruturas nos itens apresentados anteriormente.

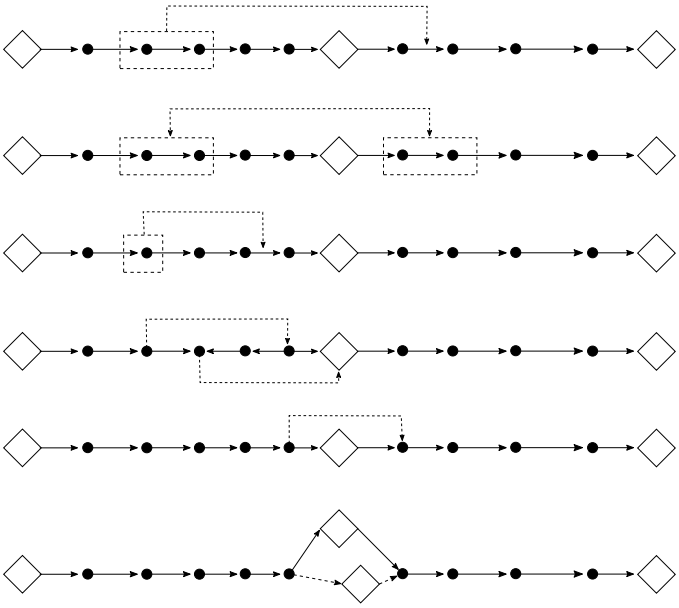

Figura 2. Estruturas de vizinhanças utilizadas na abordagem CP-VNS.

D. Variable Neighborhood Descent

A heurística conhecida como Descida de Vizinhança Variável [22] (do inglês *Variable Neigborhood Descent* - VND) consiste em um método de refinamento que explora o espaço de soluções realizando trocas entre diferentes estruturas de vizinhança $N = \{N_1, N_2, ..., N_r\}$. Apenas soluções de melhora da solução corrente são aceitas e a cada melhora, o procedimento é reiniciado aplicando-se a primeira estrutura de vizinhança. Sua ideia é baseada na premissa que um ótimo local utilizando uma estrutura de vizinhança não é necessariamente o mesmo, quando utilizada outra estrutura de vizinhança [13].

Como em alguns casos uma solução pode ser inviável para o Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH), ao utilizar tal solução o VND deve ser capaz não somente de aceitar soluções viáveis a partir da solução inicial, mas também de melhorar uma solução inviável com o intuito de torná-la viável. Para isto, a Equação (20) é utilizada para viabilizar o aceite de soluções que diminuem a inviabilidade da melhor solução corrente. Dada uma solução de entrada y, a heurística VND tenta minimizar a seguinte função $F(y)$.

$$F(y) = \sum_{d=1}^{D}(M + temp_d) + \varphi \sum_{d=1}^{D} max(temp_d - L, 0) \quad (20)$$

A Equação (20) apresenta duas parcelas bem definidas. A primeira é responsável por somar os tempos necessários para percorrer cada viagem, acrescendo-se para cada viagem o valor de uma constante M com valor suficientemente grande para priorizar soluções que contenham um número menor de viagens. A segunda realiza a soma dos tempos que excedem o limite L para cada viagem. Esta soma é multiplicada pela constante φ, fixada com um valor maior que a melhor solução conhecida na literatura. Desta forma, a equação busca a minimização do número de viagens, bem como a inviabilidade.

O VND aqui proposto considera quatro estruturas de vizinhanças N_k, $k = 1, ..., 4$, que são aplicadas na seguinte ordem: *relocate*, *exchange*, *change-hotels* e *join-trips*. A estrutura do VND é apresentada no Algoritmo 2 e consiste basicamente em um laço de repetição que é executado enquanto pelo menos uma das estruturas de vizinhança estiverem atuando positivamente sobre a solução fornecida.

Algoritmo 2: VND(y)

```
1  início
2      k ← 1
3      enquanto (k < 4) faça
4          y' ← BUSCA(N_k, y)
5          se (y' melhor que y) então
6              y ← y'
7              k ← 1
8          senão
9              k ← k + 1
10         fim
11     fim
12     Retornar y
13 fim
```

É importante observar que internamente o VND realiza a chamada para o Algoritmo 3 definido como BUSCA. É neste algoritmo que estão as chamadas para as estruturas de exploração da vizinhança. O algoritmo analisa qual é o melhor movimento para k-ésima estrutura de vizinhança aplicada a solução y, considerando a função descrita na Equação (20). Toda vez que uma solução melhor é encontrada, a nova solução é aprimorada pelos operadores *intra-trips* 2-Opt e Or-Opt. A busca é repetida até que nenhuma solução melhor seja encontrada.

Algoritmo 3: BUSCA(N_k, y)

```
1  início
2      y' ← y
3      repita
4          y'' ← argmin_{S∈N_k(y')} F(S)
5          se (y'' melhor que y') então
6              y' ← otimizar y'' com 2-OPT/OR-OPT
7          fim
8      até que nenhuma melhora for encontrada;
9      Retornar y';
10 fim
```

V. Resultados Computacionais

Nesta seção são apresentados os resultados obtidos pela abordagem proposta (CP-VNS), para cada um dos quatro grupos de problemas-teste disponíveis na literatura. As instâncias contemplam distintos níveis de dificuldade [3].

O primeiro grupo de problemas-teste aqui definido como (SET_1) foi derivado de instâncias do Problema de Rotea-

mento de Veículos Capacitado com Janela de Tempo (PRV-CJT) de Solomon *et al.* [31]. O segundo grupo definido como (SET_2) é composto por problemas-teste considerados pequenos, que foram subdivididos considerando c clientes, sendo $c = \{10, 15, 30, 40\}$. Este segundo grupo foi baseado em problemas-teste do SET_1, sendo os problemas-teste compostos pelos primeiros c_i clientes dos problemas-teste aos quais foram derivados. O terceiro grupo (SET_3) foi criado a partir de problemas-teste do clássico Problema do Caixeiro Viajante, para as quais a solução ótima é conhecida. Seus problemas-teste são subdivididos em três subgrupos que possuem respectivamente 3, 5 e 10 hotéis extras. Por fim, o quarto grupo (SET_4) é derivado do SET_3, porém para cada problemas-teste foi imposto um tempo limite arbitrário e os hotéis disponíveis foram dispostos de forma aleatória. Com isto, os problemas-teste tornaram-se mais difíceis de serem tratados, sendo seus valores ótimos desconhecidos. Os grupos de problemas-teste estão disponíveis em http://antor.ua.ac.be/tsphs.

Os experimentos foram realizados em uma máquina com processador Intel Core 2 Quad com 2.40 GHz e 4 GB de memória RAM. Com o objetivo de promover uma comparação mais justa do tempo computacional, visto que as abordagens aqui comparadas foram executadas em máquinas com configurações diferentes, os valores encontrados foram re-escalados de acordo com o procedimento descrito em [32] multiplicando-os por uma taxa igual a 0,3942.

O algoritmo CP-VNS, diferente do PLS, utiliza perturbação apenas no nível dos clientes e esta perturbação é um parâmetro definido previamente. Assim, o CP-VNS necessita de apenas dois parâmetros de entrada que são o número de iterações sem melhora e o percentual de perturbação dos clientes. Os parâmetros foram definidos empiricamente de acordo com exaustivos testes computacionais realizados com diferentes configurações. Desta forma, os valores para cada um dos parâmetros segue: i_{max} = 40 e θ_1 = 9.

Para avaliação dos resultados obtidos, foi utilizada a métrica de diferença percentual entre a melhor solução encontrada pela abordagem CP-VNS durante 30 execuções e o melhor resultado contido nos trabalhos de Castro et al. [4] e Castro et al. [13] definido como Melhor Solução da Literatura (MSL), sendo a MSL oriunda de abordagens heurísticas ou exatas. A equação que define o Gap (em porcentagem) é a seguinte:

$$Gap = 100 \times \frac{Tempo(CP-VNS) - Tempo(MSL)}{Tempo(MSL)}.$$

A fórmula para o cálculo do Gap leva em consideração apenas o tempo total da rota (incluindo o tempo de visita aos clientes), pois a quantidade de problemas-teste com o número de viagens diferentes comparando a MSL com o CP-VNS é de 6,87% para problemas-teste com acréscimo de viagens e de 1,53% para problemas-teste com redução do número de viagens. Para todas as tabelas subsequentes, valores em negrito representam as melhores soluções encontradas e os símbolos de mais (+) e menos (-) representam que houve um acréscimo ou redução no número de viagens em relação a melhor solução conhecida até o momento.

A formulação matemática foi executada e conseguiu resolver com otimalidade apenas problemas-teste com no máximo 40 clientes. Logo, o foco dos resultados serão concentrados na heurística CP-VNS que apresenta bons resultados para todos os tamanhos de problemas-teste.

Os resultados para o SET_1, que contém 16 problemas-teste, são apresentados na Tabela II. A primeira coluna contém o nome de cada problemas-teste, a segunda e terceira colunas representam o número de viagens necessárias e o tempo total da solução para a Melhor Solução da Literatura. As próximas colunas representam respectivamente o número de viagens, o tempo total necessário para realizar o percurso, o tempo computacional demandado e o Gap para a abordagem PLS e para a CP-VNS aqui proposta. É possível verificar que para seis instâncias a abordagem CP-VNS encontra as melhores soluções conhecidas na literatura, sendo que para uma delas consegue diminuir o tempo total da rota. Para o problema-teste *pr05* similarmente ao PLS, o CP-VNS também necessita de uma viagem a mais para percorrer a rota.

Adicionalmente, a Tabela I apresenta um resumo dos resultados para o SET_1, que mostram que a abordagem proposta consegue baixar o Gap médio quase pela metade em relação à MSL. Os Gaps mínimo e máximo são sempre menores em relação ao PLS. O tempo computacional médio do CP-VNS se mantém muito próximo ao demandado pelo PLS, reforçando a competitividade da abordagem proposta.

Tabela I. RESUMO RESULTADOS PARA O SET_1

	MA	PLS	CP-VNS
Melhores conhecidas	16/16	3/16	5/16
Novas Melhorias	-	-	1
Min. Gap(%)	-	0.01	-0.04
Max. Gap(%)	-	1.16	0.52
Média. Gap(%)	-	0.34	0.18
Média. Tempo Computacional	108.0	0.5	0.9

Para o grupo de problemas-teste SET_2 que contém um total de 52 problemas-teste, foram criados quatro subgrupos cada um contendo 13 problemas-teste e seus respectivos resultados são apresentados nas Tabelas III, IV, V e VI (para um número de clientes igual a 10, 15, 30 e 40). Para este grupo, apenas para cinco problemas-teste (c101 e rc101 com 30 clientes e c101, r101 e rc101 com 40 clientes) o tempo total ótimo da rota não é conhecido. Nestas tabelas, as colunas apresentam as mesmas informações conforme descrito para a Tabela II. Para os problemas-teste com 10 clientes (Tabela III), o algoritmo proposto encontrou todas as soluções ótimas, de forma similar ao PLS. Para o conjunto de problemas-teste com 15 clientes o CP-VNS encontra quase todas as soluções ótimas conhecidas mantendo um Gap médio de apenas 0,01% e tempo computacional similar ao PLS.

Uma condição especial foi encontrada para o problemas-teste *rc101* com 30 e 40 clientes que apresentou um alto valor do Gap, além de produzir rotas com acréscimo de viagens. Isto ocorre porque a melhor solução conhecida na literatura (MSL) para esta instância foi encontrada por uma abordagem exata a qual explora de forma mais efetiva o espaço de soluções possíveis. Apesar do CP-VNS também acrescer uma viagem à rota, ainda assim, consegue ser melhor que o PLS que para o problema-teste com 40 clientes necessita de duas rotas a mais.

As Tabelas V e VI sumarizam os resultados encontrados para os problemas-teste com 30 e 40 clientes, respectivamente. As soluções para o PLS e CP-VNS são similares para os

Tabela II. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET-1

Problema-teste	MSL		PLS				CP-VNS			
	V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
c101	9	9595,6	9	9596,9	0,1	0,01	**9**	**9592,1**	**0,5**	**-0,04**
r101	**8**	**1704,6**	8	1717,4	0,2	0,75	8	1713,4	0,2	0,52
rc101	**8**	**1674,1**	8	1674,3	0,2	0,01	**8**	**1674,1**	**0,1**	**0,00**
c201	**3**	**9560,0**	3	9563,1	0,1	0,03	3	9561,8	0,2	0,02
r201	**2**	**1643,4**	2	1648,1	0,1	0,29	2	1647,2	0,2	0,23
rc201	**2**	**1642,7**	2	1644,3	0,2	0,10	**2**	**1642,7**	**0,1**	**0,00**
pr01	**2**	**1412,2**	**2**	**1412,2**	**0,0**	**0,00**	**2**	**1412,2**	**0,0**	**0,00**
pr02	**3**	**2543,3**	3	2551,3	0,2	0,31	3	2548,8	0,2	0,22
pr03	**4**	**3415,1**	4	3421,1	0,3	0,18	4	3432,2	0,3	0,50
pr04	**5**	**4217,4**	**5**	**4217,4**	**0,6**	**0,00**	**5**	**4217,4**	**0,6**	**0,00**
pr05	**5**	**4958,7**	6^{+}	4974,7	1,1	0,32	6^{+}	4970,7	3,5	0,24
pr06	**7**	**5963,1**	7	6032,0	1,6	1,16	7	5982,0	4,3	0,32
pr07	**3**	**2070,3**	**3**	**2070,3**	**0,0**	**0,00**	**3**	**2070,3**	**0,0**	**0,00**
pr08	**4**	**3372,0**	4	3399,9	0,4	0,83	4	3381,1	0,7	0,27
pr09	**5**	**4420,3**	5	4445,7	1,1	0,57	5	4435,4	1,2	0,34
pr10	**7**	**5940,5**	7	5991,5	2,4	0,86	7	5954,4	2,2	0,23
Média					0,5	0,34			0,9	0,18

Tabela III. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_2 COM 10 CLIENTES

Problema-teste	MSL		PLS				CP-VNS			
	V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
c101	**1**	**955,1**	**1**	**955,1**	**0,0**	**0,00**	**1**	**955,1**	**0,0**	**0,00**
r101	**2**	**272,8**	**2**	**272,8**	**0,0**	**0,00**	**2**	**272,8**	**0,0**	**0,00**
rc101	**1**	**237,5**	**1**	**237,5**	**0,0**	**0,00**	**1**	**237,5**	**0,0**	**0,00**
pr01	**1**	**426,6**	**1**	**426,6**	**0,0**	**0,00**	**1**	**426,6**	**0,0**	**0,00**
pr02	**1**	**661,9**	**1**	**661,9**	**0,0**	**0,00**	**1**	**661,9**	**0,0**	**0,00**
pr03	**1**	**553,3**	**1**	**553,3**	**0,0**	**0,00**	**1**	**553,3**	**0,0**	**0,00**
pr04	**1**	**476,4**	**1**	**476,4**	**0,0**	**0,00**	**1**	**476,4**	**0,0**	**0,00**
pr05	**1**	**528,9**	**1**	**528,9**	**0,0**	**0,00**	**1**	**528,9**	**0,0**	**0,00**
pr06	**1**	**597,4**	**1**	**597,4**	**0,0**	**0,00**	**1**	**597,4**	**0,0**	**0,00**
pr07	**1**	**670,2**	**1**	**670,2**	**0,0**	**0,00**	**1**	**670,2**	**0,0**	**0,00**
pr08	**1**	**573,4**	**1**	**573,4**	**0,0**	**0,00**	**1**	**573,4**	**0,0**	**0,00**
pr09	**1**	**645,5**	**1**	**645,5**	**0,0**	**0,00**	**1**	**645,5**	**0,0**	**0,00**
pr10	**1**	**461,5**	**1**	**461,5**	**0,0**	**0,00**	**1**	**461,5**	**0,0**	**0,00**
Média					0,0	0,00			0,0	0,00

Tabela IV. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_2 COM 15 CLIENTES

Problema-teste	MSL		PLS				CP-VNS			
	V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
c101	**2**	**1452,2**	**2**	**1452,2**	**0,0**	**0,00**	**2**	**1452,2**	**0,0**	**0,00**
r101	**2**	**379,8**	**2**	**379,8**	**0,0**	**0,00**	**2**	**379,8**	**0,0**	**0,00**
rc101	**2**	**303,2**	**2**	**303,2**	**0,0**	**0,00**	**2**	**303,2**	**0,0**	**0,00**
pr01	**1**	**590,4**	**1**	**590,4**	**0,0**	**0,00**	**1**	**590,4**	**0,0**	**0,00**
pr02	**1**	**745,6**	**1**	**745,6**	**0,0**	**0,00**	**1**	**745,6**	**0,0**	**0,00**
pr03	**1**	**632,9**	**1**	**632,9**	**0,0**	**0,00**	**1**	**632,9**	**0,0**	**0,00**
pr04	**1**	**683,4**	**1**	**683,4**	**0,0**	**0,00**	**1**	**683,4**	**0,0**	**0,00**
pr05	**1**	**621,2**	**1**	**621,2**	**0,0**	**0,00**	1	621,4	0,0	0,03
pr06	**1**	**685,2**	**1**	**685,2**	**0,0**	**0,00**	**1**	**685,2**	**0,0**	**0,00**
pr07	**1**	**795,3**	**1**	**795,3**	**0,0**	**0,00**	**1**	**795,3**	**0,0**	**0,00**
pr08	**1**	**707,2**	**1**	**707,2**	**0,0**	**0,00**	**1**	**707,2**	**0,0**	**0,00**
pr09	**1**	**771,7**	**1**	**771,7**	**0,0**	**0,00**	1	772,9	0,0	0,16
pr10	**1**	**611,9**	**1**	**611,9**	**0,0**	**0,00**	**1**	**611,9**	**0,0**	**0,00**
Média					0,0	0,0			0,0	0,01

problemas-teste com 30 clientes, sendo o Gap médio e o tempo computacional médio iguais. Com o aumento do número de clientes (40 clientes) a abordagem proposta volta a mostrar sua competência para encontrar soluções melhores que o PLS. Esta melhora é comprovada, pois o Gap médio observado na Tabela VI para o CP-VNS é 0.34% menor do que para o PLS e os tempos computacionais médios são iguais.

Uma representação resumida dos resultados encontrados para o grupo SET_2 é apresentada na Tabela VII. Nesta tabela é possível verificar que o CP-VNS consegue valores médios de Gap sempre menores ou iguais ao PLS, mesmo encontrando um número menor de melhores soluções conhecidas (duas soluções a menos). Quanto ao tempo computacional as abordagens são similares. Adicionalmente, pode ser visto que a metaheurística que encontra o maior número de melhores soluções conhecidas é o Algoritmo Memético (MA) de Castro et al. [4]. Entretanto, para problemas-teste com 40 clientes é observado que o tempo médio do MA começa a crescer exponencialmente em relação as demais abordagens comparadas.

Totalizando 48 instâncias, o grupo SET_3 é subdividido em três grupos de 16 problemas-teste cada. Os subgrupos se diferem por apresentar um número de hotéis extras igual

Tabela V. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_2 COM 30 CLIENTES

Problema-teste	MSL		PLS				CP-VNS			
	V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
c101	**3**	**2829,4**	3	2863,6	0,0	1,21	3	2863,2	0,0	1,19
r101	**3**	**655,2**	**3**	**655,2**	**0,0**	**0,00**	**3**	**655,2**	**0,0**	**0,00**
rc101	**3**	**610,0**	4^{+}	683,8	0,0	12,10	4^{+}	683,8	0,0	12,10
pr01	**1**	**964,8**	**1**	**964,8**	**0,0**	**0,00**	**1**	**964,8**	**0,0**	**0,00**
pr02	**2**	**1078,3**	**2**	**1078,3**	**0,0**	**0,00**	**2**	**1078,3**	**0,0**	**0,00**
pr03	**1**	**952,5**	**1**	**952,5**	**0,0**	**0,00**	**1**	**952,5**	**0,0**	**0,00**
pr04	**2**	**1091,6**	**2**	**1091,6**	**0,0**	**0,00**	**2**	**1091,6**	**0,0**	**0,00**
pr05	**1**	**924,7**	**1**	**924,7**	**0,0**	**0,00**	**1**	**924,7**	**0,0**	**0,00**
pr06	**2**	**1063,2**	**2**	**1063,2**	**0,0**	**0,00**	**2**	**1063,2**	**0,0**	**0,00**
pr07	**2**	**1130,4**	**2**	**1130,4**	**0,0**	**0,00**	**2**	**1130,4**	**0,0**	**0,00**
pr08	**2**	**1006,2**	**2**	**1006,2**	**0,0**	**0,00**	**2**	**1006,2**	**0,0**	**0,00**
pr09	**2**	**1091,4**	**2**	**1091,4**	**0,0**	**0,00**	**2**	**1091,4**	**0,0**	**0,00**
pr10	**1**	**918,9**	**1**	**918,9**	**0,0**	**0,00**	**1**	**918,9**	**0,0**	**0,00**
Média					0,0	1,02			0,0	1,02

Tabela VI. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_2 COM 40 CLIENTES

Problema-teste	MSL		PLS				CP-VNS			
	V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
c101	**4**	**3817,4**	4	3866,1	0,0	1,28	4	3866,1	0,0	1,28
r101	**4**	**842,9**	4	873,5	0,0	3,63	4	862,8	0,0	2,36
rc101	**3**	**652,1**	5^{++}	870,8	0,0	33,54	4^{+}	850,3	0,0	30,39
pr01	**2**	**1160,5**	**2**	**1160,5**	**0,0**	**0,00**	**2**	**1160,5**	**0,0**	**0,00**
pr02	**2**	**1336,9**	**2**	**1336,9**	**0,0**	**0,00**	**2**	**1336,9**	**0,0**	**0,00**
pr03	**2**	**1303,4**	**2**	**1303,4**	**0,0**	**0,00**	**2**	**1303,4**	**0,0**	**0,00**
pr04	**2**	**1259,5**	**2**	**1259,5**	**0,0**	**0,00**	**2**	**1259,5**	**0,0**	**0,00**
pr05	2	**1200,7**	**2**	**1200,7**	**0,0**	**0,00**	**2**	**1200,7**	**0,0**	**0,00**
pr06	**2**	**1242,9**	**2**	**1242,9**	**0,0**	**0,00**	**2**	**1242,9**	**0,0**	**0,00**
pr07	**2**	**1407,0**	2	1410,3	0,0	0,23	2	1410,3	0,0	0,23
pr08	**2**	**1222,2**	**2**	**1222,2**	**0,0**	**0,00**	**2**	**1222,2**	**0,0**	**0,00**
pr09	**2**	**1284,2**	2	1284,4	0,0	0,02	2	1284,4	0,0	0,02
pr10	**2**	**1200,4**	**2**	**1200,4**	**0,0**	**0,00**	**2**	**1200,4**	**0,0**	**0,00**
Média					0,0	2,98			0,0	2,64

Tabela VII. RESUMO RESULTADOS PARA O SET_2

	MA				PLS				CP-VNS			
Clientes	10	15	30	40	10	15	30	40	10	15	30	40
Melhores conhecidas	13/13	13/13	11/13	10/13	13/13	13/13	11/13	8/13	13/13	11/13	11/13	8/13
Min. Gap(%)	0.00	0.00	1.19	1.27	0.00	0.00	1.21	0.02	0.00	0.03	1.19	0.02
Max. Gap(%)	0.00	0.00	15.65	30.39	0.00	0.00	12.10	33.54	0.00	0.16	12.10	30.39
Média Gap(%)	0.00	0.00	1.29	2.61	0.00	0.00	1.02	2.98	0.00	0.01	1.02	2.64
Média Tempo Computacional	0.0	0.0	0.0	0.53	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

a 3, 5 ou 10. O ponto chave deste grupo é que os hotéis foram intencionalmente definidos em posições (coordenadas x e y) que já contém um cliente no problema-teste, assim, a dificuldade consiste na distribuição dos hotéis de modo que seja satisfeito o tempo limite de uma viagem. As soluções ótimas para este grupo são definidas pelo tempo total da rota no Problema do Caixeiro Viajante (PCV). Na estrutura das tabelas foi adicionada uma segunda coluna denominada PCV que contém o valor ótimo da rota para o PCV. Ainda para cada abordagem são apresentadas colunas com o número de viagens utilizadas na rota, o tempo total da rota, o tempo computacional e a diferença percentual entre a solução encontrada e a melhor solução conhecida na literatura (MSL).

A Tabela VIII apresenta a comparação dos resultados encontrados por cada uma das abordagens comparadas quando aplicadas ao grupo SET_3 com 3 hotéis extras e é possível verificar que a abordagem CP-VNS só não consegue uma solução melhor ou igual a MSL para um dos 16 problemas-teste. Em relação ao tempo computacional médio demandado o CP-VNS é pouco mais de três vezes mais lento quando comparado ao PLS. De forma similar, na Tabela IX são apresentados os resultados para o grupo de problemas-teste que contém 5 hotéis extras. para este grupo o CP-VNS consegue consegue encontrar a melhor solução conhecida para todos os problemas-teste, sendo que ainda consegue melhorar a solução de um deles, diferente do PLS que não encontra todas as MSL. Novamente, seu tempo computacional médio é superior ao PLS, mas se mantém abaixo de 10 segundos. Finalizando os resultados para o SET_3, temos os problemas-teste que possuem 10 hotéis extras, sendo que para estes a abordagem aqui proposta consegue resultados similares ao PLS, conseguindo para um dos problemas-teste utilizar uma viagem a menos que o PLS. Quanto ao tempo computacional médio o CP-VNS é 6 vezes mais lento que que o PLS, mas este tempo não é maior que 6 segundos.

A Tabela XI ilustra um resumo dos resultados obtidos pelo algoritmo CP-VNS aplicado aos problemas-teste contidos no SET_3. Os resultados apresentados reforçam a qualidade competitiva do algoritmo proposto com as melhores abordagens contidas na literatura. Em comparação com o PLS, o CP-VNS encontra três soluções melhores a mais e adicionalmente melhora o tempo total de duas instâncias, uma para instância

Tabela VIII. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_3 COM 3 HOTÉIS EXTRAS

Problema-teste	PCV	MSL		PLS				CP-VNS			
		V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
eil51	426	**4**	**426**	**4**	**426**	**0,0**	**0,00**	**4**	**426**	**0,0**	**0,00**
berlin52	7542	**4**	**7542**	**4**	**7542**	**0,0**	**0,00**	**4**	**7542**	**0,0**	**0,00**
st70	675	**4**	**675**	**4**	**675**	**0,0**	**0,00**	**4**	**675**	**0,0**	**0,00**
eil76	538	**4**	**538**	5^+	556	0,0	3,35	5^+	556	0,0	3,35
pr76	108159	**4**	**108159**	**4**	**108159**	**0,1**	**0,00**	**4**	**108159**	**0,0**	**0,00**
kroA100	21282	**4**	**21282**	**4**	**21282**	**0,1**	**0,00**	**4**	**21282**	**0,1**	**0,00**
kroC100	20749	**4**	**20749**	**4**	**20749**	**0,0**	**0,00**	**4**	**20749**	**0,1**	**0,00**
kroD100	21294	**4**	**21294**	**4**	**21294**	**0,1**	**0,00**	**4**	**21294**	**0,1**	**0,00**
rd100	7910	**4**	**7910**	**4**	**7910**	**0,1**	**0,00**	**4**	**7910**	**0,1**	**0,00**
eil101	629	**4**	**629**	**4**	**629**	**0,1**	**0,00**	**4**	**629**	**0,1**	**0,00**
lin105	14379	**4**	**14379**	**4**	**14379**	**0,1**	**0,00**	**4**	**14379**	**0,1**	**0,00**
ch150	6528	**4**	**6528**	**4**	**6528**	**0,2**	**0,00**	**4**	**6528**	**0,2**	**0,00**
tsp225	3916	**4**	**3916**	**4**	**3916**	**0,5**	**0,00**	**4**	**3916**	**0,6**	**0,00**
a280	2579	5	2591	5	2615	0,8	0,93	$\mathbf{4}^-$	**2588**	**3,2**	**-0,12**
pcb442	50778	**4**	**50778**	5^+	51144	3,7	0,72	**4**	**50778**	**9,0**	**0,00**
pr1002	259045	**4**	**259045**	**4**	**259045**	**34,5**	**0,00**	**4**	**259045**	**139,9**	**0,00**
Média						2,5	0,31			9,6	0,20

Tabela IX. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_3 COM 5 HOTÉIS EXTRAS

Problema-teste	PCV	MSL		PLS				CP-VNS			
		V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
eil51	426	**6**	**426**	**6**	**426**	**0,0**	**0,00**	**6**	**426**	**0,0**	**0,00**
berlin52	7542	**6**	**7542**	**6**	**7542**	**0,0**	**0,00**	**6**	**7542**	**0,0**	**0,00**
st70	675	**6**	**675**	**6**	**675**	**0,0**	**0,00**	**6**	**675**	**0,0**	**0,00**
eil76	538	**6**	**538**	6	566	0,1	5,20	**6**	**538**	**0,0**	**0,00**
pr76	108159	**6**	**108159**	**6**	**108159**	**0,1**	**0,00**	**6**	**108159**	**0,0**	**0,00**
kroA100	21282	**6**	**21282**	**6**	**21282**	**0,1**	**0,00**	**6**	**21282**	**0,1**	**0,00**
kroC100	20749	**6**	**20749**	**6**	**20749**	**0,1**	**0,00**	**6**	**20749**	**0,1**	**0,00**
kroD100	21294	**6**	**21294**	**6**	**21294**	**0,1**	**0,00**	**6**	**21294**	**0,1**	**0,00**
rd100	7910	**6**	**7910**	**6**	**7910**	**0,1**	**0,00**	**6**	**7910**	**0,0**	**0,00**
eil101	629	**6**	**629**	**6**	**629**	**0,1**	**0,00**	**6**	**629**	**0,1**	**0,00**
lin105	14379	**6**	**14379**	**6**	**14379**	**0,1**	**0,00**	**6**	**14379**	**0,1**	**0,00**
ch150	6528	**6**	**6528**	**6**	**6528**	**0,2**	**0,00**	**6**	**6528**	**0,2**	**0,00**
tsp225	3916	**6**	**3916**	**6**	**3916**	**0,4**	**0,00**	**6**	**3916**	**0,6**	**0,00**
a280	2579	7	2646	7	2652	0,7	0,23	**7**	**2624**	**1,9**	**-0,83**
pcb442	50778	**6**	**50778**	7^+	51087	3,1	0,61	**6**	**50778**	**7,3**	**0,00**
pr1002	259045	**6**	**259045**	**6**	**259045**	**20,5**	**0,00**	**6**	**259045**	**114,4**	**0,00**
Média						1,6	0,38			7,8	-0,05

Tabela X. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_3 COM 10 HOTÉIS EXTRAS

Porblema-teste	PCV	MSL		PLS				CP-VNS			
		V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
eil51	426	**10**	**426**	**10**	**426**	**0,0**	**0,00**	**10**	**426**	**0,0**	**0,00**
berlin52	7542	**8**	**7864**	9^+	7542	0,0	-4,09	9^+	7542	0,0	-4,09
st70	675	**10**	**675**	**10**	**675**	**0,0**	**0,00**	**10**	**675**	**0,0**	**0,00**
eil76	538	**11**	**538**	12^+	567	0,1	5,39	12^+	565	0,1	5,02
pr76	108159	**11**	**108159**	**11**	**108159**	**0,1**	**0,00**	**11**	**108159**	**0,0**	**0,00**
kroA100	21282	**11**	**21282**	**11**	**21282**	**0,1**	**0,00**	**11**	**21282**	**0,1**	**0,00**
kroC100	20749	**11**	**20749**	**11**	**20749**	**0,1**	**0,00**	**11**	**20749**	**0,1**	**0,00**
kroD100	21294	**11**	**21294**	**11**	**21294**	**0,1**	**0,00**	**11**	**21294**	**0,1**	**0,00**
rd100	7910	**10**	**7910**	**10**	**7910**	**0,1**	**0,00**	**10**	**7910**	**0,1**	**0,00**
eil101	629	**11**	**629**	**11**	**629**	**0,1**	**0,00**	**11**	**629**	**0,1**	**0,00**
lin105	14379	**10**	**14379**	**10**	**14379**	**0,1**	**0,00**	**10**	**14379**	**0,1**	**0,00**
ch150	6528	**11**	**6528**	**11**	**6528**	**0,2**	**0,00**	**11**	**6528**	**0,2**	**0,00**
tsp225	3916	**11**	**3916**	**11**	**3916**	**0,4**	**0,00**	**11**	**3916**	**0,7**	**0,00**
a280	2579	**11**	**2613**	12^+	2596	0,7	-0,65	11	2621	1,3	0,31
pcb442	50778	**11**	**51774**	12^+	50919	2,6	-1,65	12^+	50919	6,7	-1,65
pr1002	259045	**11**	**259045**	**11**	**259045**	**11,9**	**0,00**	**11**	**259045**	**87,0**	**0,00**
Média						1,0	-0,06			6,0	-0,03

Tabela XI. RESUMO RESULTADOS PARA O SET_3

	MA			PLS			CP-VNS		
Hotéis Extras	3	5	10	3	5	10	3	5	10
Melhores conhecidas	15/16	14/16	13/16	13/16	13/16	13/16	14/16	15/16	13/16
Novas Melhorias	-	-	-	-	-	-	1	1	0
Min. Gap(%)	0.46	0.28	1.31	0.72	0.61	0.28	0.35	1.74	0.28
Max. Gap(%)	0.46	2.59	1.96	3.35	5.20	5.39	3.35	1.74	5.02
Média. Gap(%)	0.02	0.18	0.21	0.34	0.54	0.40	0.23	0.11	0.43
Média Tempo Computacional	272.1	300.3	256.4	2.5	1.6	1.0	9.6	7.8	6.0

Tabela XII. RESULTADOS PARA PROBLEMAS-TESTE DO GRUPO SET_4

Problema-teste	MSL		PLS				CP-VNS			
	V	T. Total	V	T. Total	Tempo	GAP	V	T. Total	Tempo	GAP
eil51	**6**	**429**	6	436	0,0	1,63	6	436	0,0	1,63
berlin52	**7**	**8642**	**7**	**8642**	**0,0**	**0,00**	**7**	**8642**	**0,0**	**0,00**
st70	**6**	**723**	6	731	0,0	1,11	7^+	719	0,0	-0,55
eil76	**6**	**539**	**6**	**539**	**0,0**	**0,00**	6	550	0,0	2,04
pr76	**6**	**118061**	7^+	118719	0,1	0,56	6	119167	0,2	0,94
kroA100	**6**	**22343**	7^+	22044	0,1	-1,34	7^+	21833	0,1	-2,28
kroC100	**6**	**20933**	6	21116	0,1	0,87	**6**	**20933**	**0,2**	**0,00**
kroD100	**6**	**21464**	**6**	**21464**	**0,1**	**0,00**	6	22013	0,1	2,56
rd100	**6**	**8244**	7^+	8245	0,1	0,01	6	8533	0,1	3,51
eil101	**6**	**634**	6	652	0,1	2,84	6	639	0,1	0,79
ch150	6	6647	6	6728	0,2	1,22	**6**	**6626**	**0,7**	**-0,32**
tsp225	6	4502	6	4502	0,9	0,00	**6**	**4465**	**2,7**	**-0,82**
a280	**6**	**2646**	6	2658	0,9	0,45	6	2684	1,9	1,44
pcb442	6	54339	6	55134	5,7	1,46	**6**	**54205**	**48,8**	**-0,25**
pr1002	7	290110	7	290110	29,5	0,00	$\mathbf{6}^-$	**292588**	**232,9**	**0,85**
média					2,5	0,59			19,2	0,64

com 3 hotéis extras (a280) e outra para instância com 5 hotéis extras (a280). Os três algoritmos comparados são efetivos e mantêm um gap médio abaixo de 1%. O Gap médio do CP-VNS em comparação ao PLS só não é menor para os problemas-teste com 10 hotéis extras, mesmo assim a diferença entre os Gaps é de apenas 0.03%. O algoritmo CP-VNS consegue encontrar o mesmo número de melhores soluções do MA que até o momento era o melhor algoritmo em relação a qualidade da solução. Estes testes evidenciam que para explorar melhores soluções, o algoritmo será penalizado com um tempo de execução sobressalente. Ainda assim fica claro que o CP-VNS é superior ao P-LS, mantendo a mesma ordem de grandeza em relação ao tempo computacional.

Tabela XIII. RESUMO RESULTADOS PARA O SET_4

	MA	PLS	CP-VNS
Melhores conhecidas	11/15	5/15	6/16
Novas Melhorias	-	-	4
Min. Gap(%)	-	-1.34	-2.28
Max. Gap(%)	-	2.84	3.51
Média Gap(%)	-	0.59	0.64
Média Tempo Computacional	317.4	2.59	19.2

Para o SET_4 que contém 15 problemas-teste e os resultados ótimos não são conhecidos, seus detalhes são apresentados na Tabela XII e também de forma resumida na Tabela XIII. A abordagem que produz a maior quantidade de melhores resultados é o Algoritmo Memético, porém seu tempo computacional é várias vezes maior que do PLS e CP-VNS. Entre as duas abordagens mais rápidas computacionalmente, o PLS consegue encontrar cinco das melhores soluções conhecidas, enquanto o CP-VNS encontra seis, sendo quatro delas melhores que as contidas na literatura acerca do PCVSH. O Gap mínimo do PLS e do CP-VNS aparece com valores negativos, o que indica que o tempo total da rota é menor que o melhor conhecido, porém neste caso as soluções não são melhores pois necessitam de um acréscimo no número de viagens. Para este grupo o CP-VNS obtém um Gap máximo superior ao PLS, o que indica uma desvantagem que é quase imperceptível quando comparamos o Gap médio (diferença de 0.04%). Em relação ao tempo computacional médio o CP-VNS demanda um tempo maior, porém competitivo em relação ao PLS, se observada a melhor qualidade de soluções que a primeira abordagem gera.

VI. CONCLUSÕES

O Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) é um problema que foi recentemente criado na literatura, sendo uma variante do clássico Problema do Caixeiro Viajante (PCV). O PCVSH de forma similar ao PCV é um problema pertencente à classe $\mathcal{NP}$-Difícil e possui diversas aplicações práticas.

Na literatura, existem três trabalhos que fazem uso de metaheurísticas para tratar o problema, o primeiro utiliza uma abordagem Iterated Local Search (I2LS), o segundo traz uma abordagem híbrida baseada em Algoritmo Memético (MA) combinada com Busca Tabu e, por último, uma abordagem que combina dois procedimentos de perturbação com uma metaheurística Iterated Local Search (PLS). As estratégias MA e PLS produzem soluções de boa qualidade, sendo que o MA demanda muito tempo computacional. O PLS, por sua vez, demanda baixo tempo computacional, o que é favorável para aplicações que necessitam de respostas em tempo real.

Neste trabalho, foi proposta uma formulação matemática alternativa e uma nova técnica heurística (CP-VNS). O CP-VNS combina um eficiente método de construção da solução inicial (heurística de Lin-Kernighan), com um procedimento de perturbação de clientes e um método eficaz de exploração das soluções a partir de diferentes vizinhanças (VNS). A formulação matemática é restrita a problemas-teste de pequeno porte, com um comportamento similar ao dos modelos propostos anteriormente na literatura. Contornando o problema do tamanho do problema-teste, a heurística CP-VNS consegue encontrar soluções qualitativamente melhores que o PLS e similares ao MA, com um baixo tempo computacional. Desta forma, a técnica desenvolvida se mostra competitiva com as abordagens contidas na literatura.

Trabalhos futuros podem ser conduzidos abordando extensões do problema que incluem janela de tempo, múltiplos caixeiros, custos de visita aos hotéis, bem como a criação de novos problemas-teste para cada tipo de problema. Explorando de forma eficiente as características especiais de cada problema, podem ser desenvolvidos novos algoritmos utilizando metaheurísticas combinadas aos modelos exatos de programação matemática.

AGRADECIMENTOS

Os autores gentilmente agradecem o apoio fornecido pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e o fornecimento de recursos computacionais por parte do Laboratório de Inteligência Computacional (LABIC - UFF) da Universidade Federal Fluminense.

REFERÊNCIAS

[1] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007.

[2] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427 – 441, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221710006065

[3] P. Vansteenwegen, W. Souffriau, and K. Sörensen, "The travelling salesperson problem with hotel selection," *Journal of the Operational Research Society*, vol. 63, no. 2, pp. 207 – 217, 2012.

[4] M. Castro, K. Sörensen, P. Vansteenwegen, and P. Goos, "A memetic algorithm for the travelling salesperson problem with hotel selection," *Computers & Operations Research*, vol. 40, no. 7, pp. 1716 – 1728, 2013.

[5] I. Coelho, P. Munhoz, L. Ochi, M. Souza, C. Bentes, and R. Farias, "An integrated cpu–gpu heuristic inspired on variable neighbourhood search for the single vehicle routing problem with deliveries and selective pickups," *International Journal of Production Research*, vol. 1, no. 1, pp. 1–18, 2015.

[6] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[7] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European journal of operational research*, vol. 130, no. 3, pp. 449 – 467, 2001.

[8] P. Moscato, C. Cotta, and A. Mendes, "Memetic algorithms," in *New optimization techniques in engineering*. Springer, 2004, pp. 53 – 85.

[9] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190 – 206, 1989.

[10] F. Glover., "Tabu search-part ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4 – 32, 1990.

[11] M. Castro, K. Sörensen, P. Vansteenwegen, and P. Goos, "A simple grasp+vnd for the travelling salesperson problem with hotel selection," University of Antwerp, Faculty of Applied Economics, BE, Tech. Rep., out 2012.

[12] M. M. Sousa and L. B. Gonçalves, "Comparação de abordagens heurísticas baseadas em algoritmo memético para o problema do caixeiro viajante com seleção de hotéis," in *Proc. XLVI Simpósio Brasileiro de Pesquisa Operacional*, Salvador, Brasil, 2014, pp. 1543–1554.

[13] M. Castro, K. Sörensen, P. Vansteenwegen, and P. Goos, "A fast metaheuristic for the travelling salesperson problem with hotel selection," *4OR*, pp. 1 – 20, 2014.

[14] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science, F. Glover and G. A. Kochenberger, Eds. Springer US, 2003, vol. 57, pp. 320–353.

[15] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, "Variable neighbourhood search: methods and applications," *4OR*, vol. 6, no. 4, pp. 319 – 360, 2008.

[16] A. Baltz, M. E. Ouali, G. Jäger, V. Sauerland, and A. Srivastav, "Exact and heuristic algorithms for the travelling salesman problem with multiple time windows and hotel selection," *Journal of the Operational Research Society*, 2014.

[17] J. A. Chisman, "The clustered traveling salesman problem," *Computers & Operations Research*, vol. 2, no. 2, pp. 115 – 119, 1975.

[18] M. Mestria, L. S. Ochi, and S. L. Martins, "Grasp with path-relinking for the symmetric euclidean clustered traveling salesman problem," *Computers & Operations Research*, vol. 40, pp. 3218 – 3229, 2013.

[19] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[20] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1 – 10, 2011.

[21] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498 – 516, 1973.

[22] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097 – 1100, 1997.

[23] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[24] C. Prins, P. Lacomme, and C. Prodhon, "Order-first split-second methods for vehicle routing problems: A review," *Transportation Research Part C: Emerging Technologies*, vol. 40, pp. 179 – 200, 2014.

[25] D. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, "Concorde tsp solver," 2006. [Online]. Available: http://www.tsp.gatech.edu/concorde

[26] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106–130, 2000.

[27] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985 – 2002, 2004.

[28] G. Laporte, M. Gendreau, J. Y. Potvin, and Semet, "Classical and modern heuristics for the vehicle routing problem," *International transactions in operational research*, vol. 7, no. 4-5, pp. 285 – 300, 2000.

[29] I. Or, "Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking," Ph.D. dissertation, Northwestern University, Evanston, Illinois, Jun. 1976.

[30] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791 – 812, 1958.

[31] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254 – 265, 1987.

[32] Y. Gajpal and P. L. Abad, "Multi-ant colony system (macs) for a vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. 196, no. 1, pp. 102 – 117, 2009.