COMPARAÇÃO DE ABORDAGENS HEURÍSTICAS BASEADAS EM ALGORITMO MEMÉTICO PARA O PROBLEMA DO CAIXEIRO VIAJANTE COM SELEÇÃO DE HOTÉIS

Marques Moreira de Sousa
Departamento de Informática – Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas, Campus da UFV, 36570-900, Viçosa, MG
marques.moreira@ufv.br

Luciana Brugiolo Gonçalves
Departamento de Informática – Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas, Campus da UFV, 36570-900, Viçosa, MG
lbrugiolo@ufv.br

RESUMO

Este trabalho apresenta uma comparação de três algoritmos que combinam características de algoritmos meméticos, em conjunto com procedimentos de busca local para tratar o Problema do Caixeiro Viajante com Seleção de Hotéis – PCVSH. O PCVSH é uma generalização do clássico Problema do Caixeiro Viajante – PCV, que é um problema de otimização muito explorado na literatura. No PCV, o objetivo é determinar um ciclo hamiltoniano de custo mínimo em um grafo onde os vértices representam os clientes a serem visitados. No PCVSH, considera-se um limite máximo para a duração da viagem, o que obriga a divisão dos clientes em dias de visitação. Desta forma, hotéis devem ser adicionados no percurso entre dois dias consecutivos. Os resultados obtidos com as heurísticas propostas foram comparados com os resultados de um Algoritmo Memético com Busca Tabu, proposto na literatura. Nos experimentos computacionais, é possível verificar a competitividade das heurísticas propostas neste trabalho.

PALAVRAS CHAVE. Problema do Caixeiro Viajante com Seleção de Hotéis, Heurísticas, Algoritmo Memético.

Área principal (Metaheurísticas, Otimização Combinatória, Logística e Transportes)

ABSTRACT

This work presents a comparison of three algorithms that combines characteristics of memetic algorithms with local search procedures to process the Travelling Salesperson Problem with Hotel Selection – TSPHS. The TSPHS is a generalization of classic Travelling Salesperson Problem – TSP, which is an optimization problem very studied. The TSP's objective is to determine a hamiltonian cycle of minimum costs in a graph where vertices represents the clients to be visited. In the TSPHS, a maximum limit is considered to duration trip that forces a division of clients in visiting days. This way, hotels should be added on route between two consecutive days. The results obtained with the proposed heuristics were compared with results of a Memetic Algorithm with Tabu Search proposed on literature. In the computational experiments is possible verify the competitiveness of proposed heuristics in this work.

KEYWORDS. Travelling Salesperson Problem with Hotel Selection. Heuristics. Memetic Algorithm.

Main area (Metaheuristics, Combinatorial Optimization, Logistics and Transports)

1. Introdução

O problema do caixeiro viajante (PCV) é sem dúvida um dos problemas melhor estudados em toda a matemática computacional (VANSTEENWEGEN *et al.*, 2012). Pode ser definido como um problema de fácil compreensão e descrição, que possui uma grande aplicabilidade no meio dos transportes.

No PCV, o caixeiro inicia sua jornada de trabalho em um ponto de partida, visita um grupo de clientes pré-definidos e volta para o ponto de onde partiu. Em situações onde há um limite de tempo imposto para a jornada de trabalho, podem ocorrer situações em que não é possível atender todos os clientes em apenas um dia de trabalho. 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. Assim, no final de cada jornada, uma decisão deve ser tomada: voltar ao ponto de partida ou ficar em um hotel que esteja localizado em sua rota de atendimento aos clientes. Este problema representa uma nova variante para o PCV e pode ser definido como Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH)(VANSTEENWEGEN *et al.*, 2012).

No Brasil, o problema pode ser aplicado ao transporte rodoviário, que atualmente estabelece um tempo máximo de viagem durante um dia de trabalho. Os autores em (VANSTEENWEGEN *et al*., 2012) introduziram na literatura o PCVSH e apresentaram algumas aplicações interessantes, sendo estas relacionadas a entrega de jornais e transporte de combustível.

Os autores apresentaram uma formulação matemática para o problema e uma heurística simples. A heurística é composta por dois procedimentos capazes de construir uma solução inicial e por um procedimento que busca a melhora da solução por meio da aplicação de diversos operadores de busca pelas vizinhanças. Dentre os operadores utilizados, existem operadores que foram projetados especificamente para o PCVSH e operadores de vizinhança que são comumente utilizados na literatura para o PCV. No trabalho, foi comparada a performance da heurística com o modelo matemático, utilizando grupos de instâncias criadas, a partir de instâncias do PCV, especialmente para o PCVSH. A heurística apresentou soluções com qualidade superior às obtidas pelo modelo matemático.

Em Castro *et al*., (2012) os autores apresentaram um procedimento para solução do PCVSH baseado numa heurística de duas fases, que consiste em utilizar a heurística *Greedy Randomized Adaptative Search Procedure* (GRASP) combinada com operadores de vizinhança para definir uma solução para o PCVSH, e posteriormente, aplicar uma heurística *Variable Neighbourhood Descent* (VND) para melhorar a solução produzida pelo GRASP. Foi constatado aumento no desempenho em relação ao trabalho de Vansteenwegen *et al*., (2012) para maioria das instâncias conhecidas.

No trabalho de Castro *et al.*, (2013) foi apresentada uma heurística que consiste em um Algoritmo Memético com uma Busca Tabu (BT) embutida. Neste algoritmo, a população inicial é definida utilizando as heurísticas de Lin Kernighan (1973) e inserção sequencial. Dois cromossomos da população são escolhidos por meio de torneio binário para cruzamento, gerando duas novas soluções. No cruzamento dos cromossomos escolhidos, considera-se apenas a troca de hotéis. Para otimizar as soluções geradas, é utilizado o procedimento de busca tabu que contém operadores de vizinhança utilizados em Castro *et al.*, (2012). Para instâncias de tamanho pequeno onde a solução ótima é conhecida, a heurística é capaz de encontrar a solução ótima e para as instâncias restantes encontra-se melhores resultados que os contidos na literatura.

Neste trabalho, propõe-se a comparação dos resultados alcançados pela reimplementação do trabalho de Castro *et al.*, (2013) com os resultados alcançados pelas heurísticas propostas, as quais combinam características de Algoritmos Meméticos (AM) com as metaheurísticas BT e VND.

A organização estrutural do artigo é feita da seguinte forma: na seção 2 é apresentada uma definição formal do problema abordado e um modelo de programação linear inteira para o PCVSH, a seção 3 descreve as heurísticas propostas, a seção 4 detalha os experimentos

computacionais realizados, e, por fim, na seção 5, são apresentadas as conclusões acerca do trabalho.

2. Definição do Problema

Para a melhor definição do problema, o termo "*viagem*" será utilizado para indicar uma sequência de clientes, iniciando e terminando em um hotel. O termo "*rota*" é utilizado para definir uma sequência completa de viagens conectadas, que obrigatoriamente devem visitar todos os clientes e retornar ao hotel de partida.

O problema investigado neste trabalho é definido em um grafo completo $G = (V, A)$ onde $V = H \cup C$, sendo H o conjunto dos hotéis, C o grupo de clientes e V o grupo de todas as localidades, e $A = \{(i, j) \mid i, j \in V, i \neq j\}$, onde a aresta (i, j) representa a ligação entre os clientes ou hotéis i e j. Cada cliente $i \in C$ requer um tempo de serviço ou tempo de visita T_i (com $T_i = 0$, para todo $i \in H$). A distância c_{ij} necessária para viajar da localidade i para j é conhecida para todos os pares de localidades. O mesmo hotel ($i = 0, i \in H$) deve iniciar e finalizar uma rota, podendo ser utilizado como um hotel intermediário entre viagens. Outras restrições são aplicadas, como: cada viagem deve iniciar e terminar em um dos hotéis disponíveis, a distância de uma viagem não pode exceder um limite L e uma viagem deve iniciar em um hotel onde a viagem prévia terminou (CASTRO *et al.*, 2013). 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 (VANSTEENWEGEN *et al.*, 2012).

O objetivo do PCVSH é minimizar o número de viagens necessárias para atender a todos os clientes e minimizar a distância total necessária para percorrer a rota.

Castro *et al.*, (2013) propuseram um modelo de programação linear inteira para o problema. Dado x_{ij}^d uma variável binária que recebe o valor 1 se, em uma viagem d, uma visita a um cliente ou hotel i é seguido pela visita a um cliente ou hotel j ou valor 0, caso contrário. A variável binária y^d recebe o valor 1 se na viagem d no mínimo um cliente ou hotel é visitado ou 0, caso contrário. Assim, y^d receberá o valor zero se nenhuma viagem for necessária no dia d. A constante D representa o número máximo de viagens contidas na solução. Com o intuito de priorizar soluções que apresentam um menor número de viagens, uma constante M com valor suficientemente grande, que multiplica o número de viagens, foi inserida na função objetivo.

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

$$s.t. \sum_{d=1}^{D} \sum_{i\in V} x_{ij}^d = 1, \quad j \in C \quad (2)$$

$$\sum_{i\in V} x_{ij}^d = \sum_{i\in V} x_{ji}^d, \quad j \in C, \; d = 1, \dots, D \quad (3)$$

$$\sum_{h\in H} \sum_{j\in V\setminus\{h\}} x_{hj}^d = y^d, \quad d = 1, \dots, D \quad (4)$$

$$\sum_{h\in H} \sum_{i\in V\setminus\{h\}} x_{ih}^d = y^d, \quad d = 1, \dots, D \quad (5)$$

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

$$\sum_{j\in V\setminus\{0\}} x_{0j}^1 = 1 \quad (7)$$

$$\sum_{i\in V\setminus\{0\}} x_{i0}^d \geq y^d - y^{d+1}, \quad d = 1, \dots, D-1 \quad (8)$$

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

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

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

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

$$\sum_{i\in\kappa} \sum_{j\in\kappa\setminus\{i\}} x_{ij}^d \leq |\kappa| - 1, \quad \kappa \subset C, 2 \leq |\kappa| \leq |C| - 1, \quad d = 1, \dots, D \quad (13)$$

$$x_{ij}^d \in \{0,1\}, \quad (i,j) \in A, d = 1, \dots, D \quad (14)$$

$$y^d \in \{0,1\}, \quad d = 1, \dots, D \quad (15)$$

Considerando o modelo matemático, a função objetivo (1) minimiza o número de viagens e a distância total. A restrição (2) garante que cada cliente será visitado pelo menos uma vez e a restrição (3) garante que haja conectividade entre cada viagem contida na rota. As restrições (4) e

(5) garantem que cada viagem inicia e termina em um dos H hotéis disponíveis. A restrição (6) impõe um limite superior no tamanho de cada viagem. As restrições (7) e (8) definem que a rota deve iniciar e terminar no hotel 0. 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) marcam uma viagem como sendo utilizada se, e somente se, há no mínimo a visita a um cliente ou a um hotel naquele dia. Restrições (12) garantem que as viagens serão executadas em dias consecutivos, iniciando no dia 1. As restrições (13) que envolvem κ subconjuntos do grupo C de clientes, definem a clássica restrição de eliminação de ciclos aplicada a cada viagem. Como uma solução viável do PCVSH pode conter ciclos iniciando e terminando no mesmo hotel, os subgrupos na restrição de eliminação de rotas internas (13) envolvem apenas clientes. Por fim, as restrições (14) e (15) indicam o domínio das variáveis.

O modelo apresentado é capaz de encontrar resultados ótimos para instâncias com número limitado a no máximo 30 clientes. A medida que o número de clientes aumenta, torna-se mais difícil para o método exato encontrar soluções relevantes em tempo computacional aceitável. Heurísticas trabalham muito bem para pequenas instâncias e são eficientes computacionalmente para instâncias com maior número de clientes. Assim, o foco deste trabalho, é a comparação de abordagens heurísticas para o PCVSH.

3. Heurísticas Propostas

Como o problema tratado neste trabalho é uma generalização do PCV, que é reconhecidamente um problema classificado como NP-Difícil (GAREY E JOHNSON, 1979), logo o PCVSH é também classificado como NP-Difícil (VANSTEENWEGEN *et al*., 2012).

Neste trabalho propõe-se duas abordagens que são compostas pela ideia geral de um Algoritmo Memético (MERZ, 2002; AMORIN *et al.*, 2012) e das metaheurísticas BT (GLOVER, 1989; GLOVER, 1990) e VND (HANSEN E MLADENOVIC, 2003). O AM é um algoritmo evolucionário que, a cada geração, procura melhorar a população inicial por meio de cruzamentos e aplicação de operadores de vizinhança, convergindo a população para melhores soluções. A BT é uma metaheurística que utiliza estruturas de vizinhança de forma adaptativa para explorar um espaço de busca e o VND é uma metaheurística baseada em sucessivas execuções de buscas locais utilizando diferentes estruturas de vizinhanças.

Nas heurísticas propostas, denominadas AM_BT e AM_RVND utilizam-se respectivamente: a estrutura de BT utilizada em Castro *et al.*, (2013) e uma variante do VND similar a utilizada no trabalho de Campos *et al.*, (2013), estratégia que vem apresentando bons resultados para problemas de roteamento (PENNA *et al.*, 2013; SUBRAMANIAN *et al.*, 2010). Na variante RVND (*Random Variable Neighborhood Descent*) a ordem de execução das estruturas de vizinhança aplicadas à solução é definida de forma aleatória a cada iteração do algoritmo.

O pseudocódigo baseado em Amorin *et. al.*, (2012) é apresentado no Algoritmo 1. O algoritmo recebe como entrada quatro parâmetros: tamanho da população (p), o número máximo de iterações sem melhora (*maxIsm*), o percentual de mutação (*mut*) e a probabilidade de busca local (*bl*). A diferença entre as abordagens está no procedimento de busca local que é aplicado.

3.1. Operações do Algoritmo Memético

Um detalhamento do funcionamento das operações observadas no Algoritmo 1 é apresentado nesta seção.

3.1.1. Métodos Construtivos

Para gerar uma solução para o Problema do Caixeiro Viajante com Seleção de Hotéis (PCVSH) são utilizados dois métodos distintos, chamados M_1 e M_2.

O primeiro método construtivo M_1define uma solução viável para o PCVSH com base em uma rota definida para o PCV, desconsiderando-se os hotéis. Nesta rota o hotel inicial e final são fixados (h_0) e todos os clientes devem obrigatoriamente ser visitados, sem levar em consideração o limite L de uma viagem. A rota PCV é encontrada por meio da aplicação da heurística de Lin e Kernighan (1973) (LKH), da forma como foi implementada em Applegate *et*

al., (2006). A rota gerada definida por $R = <h_0, c_1, ..., c_n>$, para maioria das instâncias é inviável para o PCVSH (h_0 representa o hotel inicial e final e $c_1, ..., c_n$ representam os n clientes na rota).

Algoritmo 1: Algoritmo_Memetico(*p, maxIsm, mut, bl*)	
1:	P ← populacao inicial;
2:	Aplicar 2-opt em |*P*| * 0.2;
3:	ism ← 0;
4:	**Enquanto** (ism < *maxIsm*), **faça**
5:	P' ← elite(P);
6:	Seleção: p_1 e p_2 de P;
7:	P' ← cruzamento: $p_1 \otimes p_2$;
8:	Aplicar mutação;
9:	Melhorar solução com RVND ou BT;
10:	Preencher o restante da populacao P';
11:	P ← P';
12:	**Fim_Enquanto**

Para os casos em que a rota gerada PCV não representa uma solução do PCVSH, ou seja, cuja duração ultrapassa o limite L, um procedimento de particionamento da rota em viagens viáveis, ou seja, onde L seja satisfeito, é utilizado. Este procedimento utiliza uma adaptação do algoritmo de Dijkstra (1959), que é completamente detalhado em Castro *et al.*, (2013).

O segundo método construtivo M_2 define uma solução para o PCV utilizando a heurística de inserção mais barata (TALBI, 2009). Na heurística de inserção mais barata, a rota é construída de forma iterativa. Inicialmente, a rota é composta pelo hotel inicial e final (h_0) e por dois clientes escolhidos de forma aleatória dentre os |*C*| clientes. O restante dos |*C*| - 2 clientes são inseridos em ordem aleatória na posição em que seja menor o acréscimo no tamanho total da rota, considerando todas as possibilidades de inserção na rota para cada cliente.

Após a inserção de todos os clientes à rota, ainda sem considerar o limite L, é aplicado o operador de vizinhança *3-opt* (LIN, 1965) com o objetivo de determinar uma sequência de clientes com menor tamanho total. De forma semelhante ao primeiro método construtivo, a rota PCV gerada pode ou não representar uma solução viável para o PCVSH. O procedimento de particionamento apresentado no primeiro método baseado no algoritmo de Dijkstra é então aplicado à rota gerada, criando uma rota que atende à restrição de tamanho imposta a cada viagem.

3.1.2. Geração da População Inicial

Para a geração da população inicial do AM, linha 1 do Algoritmo 1, são utilizados os dois métodos construtivos descritos na seção anterior. A cada cromossomo p_i da população P está associado uma rota R.

O primeiro membro da população é gerado utilizando o método M_1. Este membro corresponde a uma solução para o PCVSH e garante que todas as viagens estejam dentro do limite de tamanho máximo definido pela instância. A solução gerada é considerada de boa qualidade e define um bom limite superior para o número de viagens (CASTRO *et al.*, 2013).

A rota gerada passa a fazer parte da população P e o restante da população $t - 1$ deve ser gerada. Para gerar o restante da população o método M_1 não pode ser utilizado, por ser um método determinístico. Logo, os cromossomos restantes devem ser gerados utilizando o método M_2. Após aplicar o M_2, é verificado se a rota gerada contém um número maior de viagens em relação a rota gerada por M_1. Caso o número de viagens seja maior, os clientes pertencentes às rotas excedentes devem ser realocados para as viagens onde seu custo de inserção gere o menor impacto no tamanho total da rota, ou seja, onde a modificação cause menor inviabilidade. Finalmente, a 20% da população P é aplicado o procedimento de busca local *2-opt* (TALBI, 2009), linha 2 do Algoritmo 1, com o intuito de melhorar as soluções. Para compor a nova

população *P'* que será gerada, a população *P* é ordenada considerando o menor número de viagens e tamanho total da rota. A população é então dividida em três classes: Classe A ou elite (20% das melhores soluções), Classe B ou intermediária (65%) e Classe C (15% das soluções com pior qualidade).

A cada iteração do AM, linhas 4 a 12 do Algoritmo 1, uma nova população é gerada (*P'*). A classe elite da população atual *P* é clonada na nova população *P'* (linha 5). A classe C é reconstruída utilizando-se o método construtivo M_2 (linha 10). A classe intermediária é obtida a partir da seleção, cruzamento, mutação e melhoramento, linhas 6 a 9, como descrito nas próximas seções.

3.1.3. Seleção e cruzamento

Para compor a população *P'*, novos membros (65%) serão gerados por meio de cruzamentos de membros da população *P*. Para a escolha dos membros que devem participar do cruzamento, linha 6 do Algoritmo 1, é realizada a seleção aleatória de um membro da elite de *P* e outro membro que está na Classe B ou Classe C de *P*.

O cruzamento utilizado, linha 7 Algoritmo 1, realiza a troca considerando apenas os clientes, sendo os hotéis intermediários retirados das rotas antes da operação. O operador de cruzamento de dois pontos (AMORIN *et al.*, 2012) é aplicado aos membros selecionados produzindo novas soluções, o_1 e o_2, denominadas proles. Novamente, o procedimento de particionamento da rota PCV é aplicado às proles o_1 e o_2, tornando-as soluções para o PCVSH. Após realizar o cruzamento e particionamento, as novas soluções são inseridas em *P'*.

3.1.4. Mutação da Solução

Para evitar que a população fique homogênea, um operador de mutação deve ser aplicado, linha 8 do Algoritmo 1, para permitir que outras soluções também possam ser exploradas (TALBI, 2009). Um método de mutação foi desenvolvido com base nos movimentos da vizinhança *4-opt*. Este método define aleatoriamente 4 pontos na rota onde serão removidas as arestas e testa o tamanho total da rota para cada forma possível de reconectar os pontos de acordo com a estratégia *4-opt*. A forma de reconectar os pontos que apresentar o menor tamanho total é aplicada a rota. A quantidade de membros de *P'* que serão sujeitos a mutação é definida considerando o valor mínimo entre (0.2 * |*p*|) e (*mut* * |*ism*|). Para evitar que a melhor solução de *P'* seja alterada, esta não pode ser escolhida para sofrer mutação.

3.1.5. Melhoramento da População

Após a mutação, um procedimento de melhoria (RVND para AM_RVND ou BT para AM_BT), linha 9 do Algoritmo 1, é aplicado a uma porcentagem (|*p*|**bl*) dos membros de *P'*, excluindo os membros da elite. O procedimento de Busca Tabu (BT) utilizado é descrito em Castro *et al.*, (2013) e o procedimento RVND é detalhado na seção 3.2.

Após o procedimento de melhora, a população *P'* ainda não conterá os *p* membros. Para preencher a população *P'*, o método de inserção mais barata (descrito na Seção 3.1.1) é utilizado. Antes de iniciar uma nova iteração do algoritmo, a qualidade da melhor solução de *P'* é comparado com a qualidade da melhor solução de *P*, caso sejam diferentes a variável *ism* será zerada, caso contrário, será incrementada. Por fim, os membros de *P* são substituídos pelos membros de *P'* e uma nova iteração do algoritmo é realizada.

3.2. VND com Ordenação Aleatória das Vizinhanças (RVND)

No VND, a ordem de aplicação das estruturas de vizinhança é definida de acordo com algum critério de ordenação. Ao explorar a primeira estrutura de vizinhança contida na ordem pré-definida, se não houver melhora da solução, então a próxima vizinhança deve ser explorada. Caso contrário, deve-se voltar a primeira estrutura de vizinhança, executando novamente o procedimento, que somente será finalizado quando nenhuma das estrutura de vizinhança for capaz de melhorar a solução corrente. Nesta abordagem (RVND), a ordem de utilização das estruturas é definida de forma aleatória a cada vez que o procedimento é requisitado.

Com o intuito de prover um número maior de possibilidades de exploração das vizinhanças, foram utilizados 4 operadores diferentes, são eles: *Relocate*, *Exchange*, *JoinTrips* e *2-opt*. Dentre estes operadores, apenas o *2-opt* não foi utilizado por Castro *et al.*, (2013) no desenvolvimento de sua heurística.

O operador *Relocate*, remove uma cadeia de k clientes consecutivos de uma viagem e insere em outra viagem (k = 3). O *Exchange* realiza a troca de k clientes consecutivos de uma viagem com outros k clientes de outra viagem (neste operador o k assume os valores 1, 2 e 3). O *JoinTrips* percorre a rota, tentando retirar hotéis intermediários, com o objetivo de concatenar duas viagens consecutivas. Ao concatenar as viagens, o tamanho da nova viagem não pode exceder o limite de tamanho da viagem definido pela instância. Por fim, o operador *2-opt* busca reorganizar os clientes nas viagens de uma forma diferente, visando a exploração de vizinhanças e otimização da solução.

4. Experimentos Computacionais

Nesta seção, os resultados obtidos são comparados aos resultados encontrados pela reimplementação da heurística de Castro *et al.*, (2013), definida neste trabalho como AM+BTLit. As heurísticas desenvolvidas foram testadas nos grupos de instâncias contidas na literatura, totalizando 131 instâncias. Os experimentos foram executados usando um computador com sistema operacional UBUNTU 13.10, processador Intel Core i5-3570 (3.40GHz) e 16GB de RAM. O algoritmo foi codificado em C++ e compilado com o G++ versão 4.8.1.

As instâncias utilizadas para condução dos testes, consistem em quatro grupos de instâncias definidas por Vansteenwegen *et al.*, (2012). Todos os grupos foram gerados com base em instâncias utilizadas no PCV e PRV. O procedimento de geração das instâncias é detalhado em http://antor.ua.ac.be/tsphs.

O primeiro grupo (SET_1) é composto por 16 instâncias, seis delas contendo 100 clientes cada e outras dez instâncias contendo quantidades de clientes variando entre 48 e 288. O segundo grupo (SET_2), foi adaptado a partir de 13 instâncias do SET_1, usando somente os primeiros 10, 15, 30 e 40 clientes da instância original. O terceiro grupo (SET_3) possui um número de clientes definido entre 51 e 1002. Foram utilizadas 16 instâncias diferentes e atribuídos hotéis extras (3, 5 e 10), além do hotel 0. O último grupo (SET_4) foi gerado com base no SET_3, porém contém 10 hotéis disponíveis e as soluções não são conhecidas.

4.1. Resultados

Nesta seção é feita a comparação dos resultados obtidos pelas abordagens propostas, com os resultados do AM+BTLit. Para cada umas das instância, o algoritmo foi executado 10 vezes e nas comparações foram utilizados os melhores resultados, como em Castro *et al.*, (2013).

Para comparação do tempo de execução, foi considerado o tempo de CPU. Os parâmetros utilizados pelo AM+BTLit, foram definidos em Castro *et al.*, (2013) com a alteração do método de geração da solução inicial Inserção Sequencial para Inserção mais Barata. Para as heurísticas propostas, foram utilizados os parâmetros (baseados em AMORIN *et al.*, 2012): tamanho da população (t = 20), iterações sem melhora ($maxIsm$ = 10), taxa de mutação (mut = 0.02) e taxa de busca local (bl = 0.2).

Para todas as tabelas, resultados em negrito indicam a melhor solução encontrada. Os resultados para as instâncias do SET_1 são apresentados na Tabela 1. A primeira coluna representa o nome da instância. As colunas 2, 3 e 4 mostram o número de viagens (V), tamanho total da rota (Tam) e o tempo de CPU gasto na execução do algoritmo AM+BTLit. As colunas 5, 6 e 7 apresentam os resultados obtidos pela abordagem AM_BT. A coluna 8 apresenta a diferença percentual da solução obtida pelo AM_BT em relação ao AM+BTLit. As colunas 9, 10 e 11 apresentam os resultados obtidos pela abordagem AM_RVND. A coluna 12 apresenta a diferença percentual da solução obtida pelo AM_RVND em relação ao AM+BTLit. Ambas heurísticas obtiveram resultados melhores que o AM+BTLit, diminuindo em média 0,6% o tamanho das rotas. O AM_BT encontrou mais de 80% das melhores soluções para este grupo. Em relação ao tempo gasto, o AM_RVND é, em média, de 3 a 7 vezes mais rápido que os outros.

Os resultados para as instâncias do SET_2 são apresentadas nas Tabelas 2-5, para 10, 15, 30 e 40 clientes, respectivamente. As Tabelas 2 e 3 contém os resultados para instâncias com 10 e 15 clientes. Para este número de clientes as heurísticas obtiveram uma melhora percentual em média igual a 0,3%, quando comparadas com o AM+BTLit.

Tabela 1: Resultados computacionais para as instâncias do SET_1.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T(s)	GAP(%)
c101	9	9593,3	20,6	9	9587,9	26,3	-0,06	9	**9587,7**	6,3	-0,06
r101	8	1742,4	53,4	9	**1733,9**	87,3	-0,49	8	1734,2	8,0	-0,47
rc101	8	1684,1	20,7	8	**1675,6**	37,6	-0,50	8	1679,2	3,4	-0,29
c201	3	9563,7	18,1	3	**9560,4**	22,7	-0,03	3	9561,5	8,8	-0,02
r201	2	1651,0	18,0	2	**1640,8**	43,8	-0,62	2	1641,2	8,0	-0,59
rc201	2	1651,5	16,9	2	**1643,9**	23,4	-0,46	2	1645,1	5,7	-0,39
pr01	2	**1412,2**	2,2	2	**1412,2**	2,1	0,00	2	**1412,2**	0,8	0,00
pr02	3	2571,1	17,0	3	**2543,3**	26,9	-1,08	3	2551,7	4,4	-0,75
pr03	4	3456,2	53,2	4	3435,5	74,8	-0,60	4	**3432,8**	13,3	-0,68
pr04	5	4288,6	129,1	5	**4225,8**	189,0	-1,46	5	4228,1	49,2	-1,41
pr05	6	5077,9	290,8	6	**4999,4**	727,2	-1,55	6	5000,6	109,4	-1,52
pr06	7	6056,0	489,2	7	**5997,2**	969,0	-0,97	7	6032,9	132,2	-0,38
pr07	3	2082,8	7,3	3	**2070,3**	8,1	-0,60	3	**2070,3**	2,3	-0,60
pr08	4	3414,9	52,4	4	**3382,2**	73,3	-0,96	4	3385,6	14,5	-0,86
pr09	5	4488,4	177,8	5	**4442,5**	631,5	-1,02	5	4462,9	26,6	-0,57
pr10	7	6035,4	511,1	7	**5949,6**	1104,5	-1,42	7	5994,3	140,0	-0,68
Média			117,4			253,0	-0,74			33,3	-0,58

Tabela 2: Resultados computacionais para as instâncias do SET_2 com 10 clientes.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
c101	1	956,8	0,0	1	**955,1**	0,0	-0,18	1	**955,1**	0,0	-0,18
r101	2	**272,8**	0,0	2	**272,8**	0,0	0,00	2	**272,8**	0,0	0,00
rc101	1	**237,5**	0,0	1	**237,5**	0,0	0,00	1	**237,5**	0,0	0,00
pr01	1	**426,6**	0,0	1	**426,6**	0,0	0,00	1	**426,6**	0,0	0,00
pr02	1	**661,9**	0,0	1	**661,9**	0,0	0,00	1	**661,9**	0,0	0,00
pr03	1	**553,3**	0,0	1	**553,3**	0,0	0,00	1	**553,3**	0,0	0,00
pr04	1	**476,4**	0,0	1	**476,4**	0,0	0,00	1	**476,4**	0,0	0,00
pr05	1	**528,9**	0,0	1	**528,9**	0,0	0,00	1	**528,9**	0,0	0,00
pr06	1	604,1	0,0	1	**597,4**	0,0	-1,11	1	**597,4**	0,0	-1,11
pr07	1	688,9	0,0	1	**670,2**	0,0	-2,71	1	**670,2**	0,0	-2,71
pr08	1	**573,4**	0,0	1	**573,4**	0,0	0,00	1	**573,4**	0,0	0,00
pr09	1	**645,5**	0,0	1	**645,5**	0,0	0,00	1	**645,5**	0,0	0,00
pr10	1	**461,5**	0,0	1	**461,5**	0,0	0,00	1	**461,5**	0,0	0,00
Média			0,0			0,0	-0,31			0,0	-0,31

Tabela 3: Resultados computacionais para as instâncias do SET_2 com 15 clientes.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
c101	2	**1452,2**	0,1	2	**1452,2**	0,1	0,00	2	**1452,2**	0,0	0,00
r101	2	**379,8**	0,1	2	**379,8**	0,1	0,00	2	**379,8**	0,0	0,00
rc101	2	**303,2**	0,1	2	**303,2**	0,1	0,00	2	**303,2**	0,0	0,00
pr01	1	**590,4**	0,0	1	**590,4**	0,0	0,00	1	**590,4**	0,0	0,00
pr02	1	**745,6**	0,0	1	**745,6**	0,0	0,00	1	**745,6**	0,0	0,00
pr03	1	**632,9**	0,0	1	**632,9**	0,0	0,00	1	**632,9**	0,0	0,00
pr04	1	693,2	0,0	1	**683,4**	0,0	-1,41	1	**683,4**	0,0	-1,41
pr05	1	623,1	0,0	1	**621,2**	0,0	-0,30	1	**621,2**	0,0	-0,30
pr06	1	**685,2**	0,0	1	**685,2**	0,0	0,00	1	**685,2**	0,0	0,00
pr07	1	800,4	0,0	1	**795,3**	0,0	-0,64	1	**795,3**	0,0	-0,64
pr08	1	715,7	0,0	1	**707,2**	0,0	-1,19	1	**707,2**	0,0	-1,19
pr09	1	**771,7**	0,0	1	**771,7**	0,0	0,00	1	**771,7**	0,0	0,00
pr10	1	**611,9**	0,0	1	**611,9**	0,0	0,00	1	**611,9**	0,0	0,00
Média			0,0			0,0	-0,27			0,0	-0,27

As Tabelas 4 e 5 apresentam os resultados para as instâncias que possuem um número de clientes igual a 30 e 40. Para estes dois grupos as heurísticas propostas alcançaram melhores resultados. A Tabela 4, apresenta os resultados para instâncias com 30 clientes. As heurísticas propostas conseguem resultados em média 0,4 a 0,7% melhores e o AM_RVND é 5 vezes mais rápido que o AM+BTLit.

Tabela 4: Resultados computacionais para as instâncias do SET_2 com 30 clientes.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
c101	3	2868,8	0,6	3	**2866,5**	1,0	-0,08	3	2869,4	0,1	0,02
r101	4	663,3	0,6	3	656,1	2,2	-1,09	3	**655,2**	0,1	-1,22
rc101	4	683,8	0,8	4	684,4	0,6	0,09	4	**657,2**	0,1	-3,89
pr01	1	974,6	0,0	1	**964,8**	0,4	-1,01	1	**964,8**	0,0	-1,01
pr02	2	1083,1	0,7	2	**1082,9**	0,5	-0,02	2	**1082,9**	0,1	-0,02
pr03	1	972,9	0,0	1	**952,5**	0,4	-2,10	1	**952,5**	0,1	-2,10
pr04	2	**1091,6**	0,6	2	**1091,6**	0,4	0,00	2	**1091,6**	0,0	0,00
pr05	1	936,6	0,0	1	**924,7**	0,4	-1,27	1	**924,7**	0,0	-1,27
pr06	2	1065,3	0,6	2	**1063,2**	0,5	-0,20	2	1065,3	0,1	0,00
pr07	2	**1130,4**	0,7	2	**1130,4**	0,4	0,00	2	**1130,4**	0,1	0,00
pr08	2	**1006,2**	0,7	2	**1006,2**	0,3	0,00	2	**1006,2**	0,0	0,00
pr09	2	**1091,4**	0,6	2	**1091,4**	0,5	0,00	2	**1091,4**	0,0	0,00
pr10	1	**918,9**	0,0	1	**918,9**	0,4	0,00	1	**918,9**	0,0	0,00
Média			0,5			0,6	-0,44			0,1	-0,73

Os dados apresentados na Tabela 5, demonstram que o AM_BT e o AM_RVND encontraram resultados iguais ou melhores na maioria das instâncias, tendo solução diferente em apenas uma instância, porém a diferença não chega a 1%. Em relação ao tempo de CPU, novamente o AM_RVND é mais rápido em relação as outras abordagens.

Os resultados para as instâncias do SET_3 são apresentados nas Tabelas 6 - 8. A Tabela 6, detalha os resultados encontrados para as instâncias que contém 3 hotéis extras. Para estas instâncias as heurísticas propostas alcançam melhores resultados e para algumas instâncias conseguem diminuir o número de viagens utilizadas. Em relação ao tempo de CPU, as abordagens que utilizam a BT demandam uma quantia superior de tempo, quando comparadas com a abordagem que utiliza RVND. As Tabelas 7 e 8, ilustram os resultados encontrados para o SET_3 com 5 e 10 hotéis extras. Novamente, as heurísticas propostas alcançam resultados melhores que o AM+BTLit.

Tabela 5: Resultados computacionais para as instâncias do SET_2 com 40 clientes.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
c101	4	3872,3	1,4	4	**3866,1**	1,1	-0,16	4	3868,0	0,3	-0,11
r101	4	874,2	1,6	4	878,5	2,2	0,49	4	**872,2**	0,3	-0,23
rc101	4	851,1	1,3	4	**850,9**	1,0	-0,02	4	851,2	0,1	0,01
pr01	2	1165,1	1,3	2	**1160,5**	1,1	-0,39	2	**1160,5**	0,1	-0,39
pr02	2	**1336,9**	1,3	2	**1336,9**	1,0	0,00	2	**1336,9**	0,1	0,00
pr03	2	**1303,4**	1,3	2	**1303,4**	1,0	0,00	2	**1303,4**	0,1	0,00
pr04	2	**1259,5**	1,4	2	**1259,5**	1,1	0,00	2	**1259,5**	0,1	0,00
pr05	2	**1200,7**	1,5	2	**1200,7**	1,5	0,00	2	**1200,7**	0,1	0,00
pr06	2	**1242,9**	1,5	2	**1242,9**	1,0	0,00	2	**1242,9**	0,1	0,00
pr07	2	**1407,2**	1,3	2	**1407,2**	1,1	0,00	2	**1407,2**	0,1	0,00
pr08	2	**1222,2**	1,6	2	**1222,2**	1,3	0,00	2	**1222,2**	0,1	0,00
pr09	2	**1284,4**	1,3	2	**1284,4**	1,8	0,00	2	**1284,4**	0,2	0,00
pr10	2	**1200,4**	1,4	2	**1200,4**	2,6	0,00	2	**1200,4**	0,1	0,00
Média			1,4			1,4	-0,01			0,1	-0,06

Na Tabela 8, a instância berlin_52 apresenta um tamanho total da rota menor do que o encontrado pelas heurísticas propostas, porém utiliza um número maior de viagens.

Tabela 6: Resultados computacionais para as instâncias do SET_3 com 3 hotéis extras.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
eil_51	5	454	3,4	4	**435**	3,4	-4,34	4	**435**	0,6	-4,34
berlin_52	4	**7658**	3,9	4	**7658**	3,2	0,00	4	**7658**	0,3	0,00
st_70	4	676	15,1	4	**675**	7,6	-0,16	4	**675**	0,7	-0,16
eil_76	5	578	10,4	5	**555**	31,8	-3,91	5	561	1,6	-2,96
pr_76	4	108272	10,0	4	108159	10,0	-0,10	4	**108157**	1,2	-0,11
kroa_100	4	21319	28,2	4	**21307**	16,3	-0,05	4	21319	3,7	0,00
kroc_100	5	21453	18,6	4	**20749**	26,6	-3,28	4	**20749**	4,5	-3,28
krod_100	5	21732	19,6	4	**21325**	32,2	-1,87	4	21396	2,9	-1,55
rd_100	5	8314	19,6	4	**7910**	30,0	-4,86	4	**7910**	4,2	-4,86
eil_101	5	670	20,4	5	**655**	44,6	-2,25	5	656	7,9	-2,06
lin_105	5	14650	25,0	4	**14380**	21,7	-1,84	4	**14380**	2,5	-1,84
ch_150	5	6931	67,5	5	6669	76,1	-3,77	5	**6552**	12,5	-5,47
tsp_225	5	4097	188,1	5	**3991**	425,7	-2,59	5	4010	73,6	-2,14
a_280	5	2785	353,9	5	2659	1058,8	-4,52	5	**2635**	194,5	-5,39
pcb_442	5	54855	1606,6	5	52929	4704,0	-3,51	5	**52664**	506,9	-3,99
pr_1002	5	278458	43874,5	5	270958	117781,1	-2,69	5	**267165**	17435,3	-4,06
Média			2891,6			7767,1	-2,49			1140,8	-2,6

Tabela 7: Resultados computacionais para as instâncias do SET_3 com 5 hotéis extras.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
eil_51	6	446	7,7	6	**435**	4,6	-2,44	6	443	0,4	-0,52
berlin_52	6	**7543**	4,5	6	**7543**	3,3	0,00	6	**7543**	0,4	0,00
st_70	6	**675**	8,5	6	**675**	8,0	0,00	6	**675**	1,0	0,00
eil_76	7	592	10,9	6	**552**	26,4	-6,70	6	580	1,7	-2,03
pr_76	7	110134	9,9	6	**108157**	15,8	-1,80	6	**108157**	1,7	-1,80
kroa_100	7	21590	19,7	6	**21294**	26,4	-1,37	6	**21294**	5,1	-1,37
kroc_100	6	**20749**	41,4	6	**20749**	24,9	0,00	6	**20749**	4,1	0,00
krod_100	6	21450	44,3	6	**21302**	32,7	-0,69	6	21389	3,4	-0,28
rd_100	7	8208	19,8	6	**8039**	30,0	-2,06	6	8045	2,8	-1,98
eil_101	7	678	45,4	7	**663**	57,5	-2,15	7	665	7,9	-1,80
lin_105	7	14712	23,9	6	**14438**	24,4	-1,86	6	**14438**	3,2	-1,86
ch_150	8	7068	64,4	7	6649	169,6	-5,93	6	**6616**	24,1	-6,41
tsp_225	7	4145	247,6	7	**4013**	557,7	-3,18	7	4064	68,3	-1,94
a_280	8	2819	372,7	7	**2716**	798,2	-3,65	7	2741	58,8	-2,75
pcb_442	7	54519	1783,0	7	**52870**	10231,4	-3,02	7	53113	906,5	-2,58
pr_1002	7	275286	43726,8	7	272327	87987,3	-1,07	7	**271111**	27045,1	-1,52
Média			2901,9			6249,9	-2,25			1758,4	-1,68

Tabela 8: Resultados computacionais para as instâncias do SET_3 com 10 hotéis extras. Asterisco indica solução com número menor de viagens.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
eil_51	12	490	7,3	12	473	6,4	-3,41	12	**470**	1,1	-4,20
berlin_52	9	7543	3,8	8	7991	9,2	5,94	8	**7980***	1,1	5,80
st_70	11	709	17,9	11	**698**	20,2	-1,52	11	709	1,6	0,00
eil_76	13	596	13,8	12	575	18,1	-3,61	12	**571**	2,3	-4,28
pr_76	11	**115665***	26,5	12	112846	13,9	-2,44	12	114177	2,6	-1,29
kroa_100	11	21282	30,6	11	**21281**	23,0	0,00	11	**21281**	2,6	0,00
kroc_100	12	21335	23,5	12	21027	24,8	-1,45	12	**20922**	4,0	-1,94
krod_100	12	21808	31,7	11	**21294**	29,5	-2,36	11	**21294**	8,7	-2,36
rd_100	11	8248	21,6	10	**7910**	29,9	-4,10	10	**7910**	4,1	-4,10
eil_101	12	685	45,8	12	658	86,4	-3,96	11	**641**	6,7	-6,51
lin_105	10	**14391**	24,6	10	**14391**	31,7	0,00	10	**14391**	3,6	0,00
ch_150	11	7105	145,0	11	**6592**	165,0	-7,22	11	6781	17,0	-4,56
tsp_225	13	4174	209,9	12	**4056**	469,2	-2,84	12	4143	78,6	-0,75
a_280	14	2887	432,2	12	**2732**	1797,8	-5,40	12	2735	255,9	-5,27
pcb_442	13	55640	3404,0	12	**53124**	11575,5	-4,52	12	54060	727,9	-2,84
pr_1002	13	284444	48331,6	13	**274070**	213179,6	-3,65	13	275556	14985,3	-3,12
Média			3298,1			14217,5	-2,53			1006,4	-2,21

O AM+BTLit consegue um resultado melhor (considerando o número de viagens) que as outras abordagens para a instância pr_76, diminuído o número de viagens de 12 para 11. Em relação ao tempo de CPU o AM_RVND é o mais rápido para os dois grupos de instâncias.

Finalmente, a Tabela 9 apresenta os resultados para o grupo SET_4. Para duas instâncias (berlin_52 e st_70) o AM+BTLit encontrou resultados melhores que o AM_BT e AM_RVND, porém a diferença percentual da solução não chega a 1%. Para as demais instâncias as heurísticas propostas encontram soluções iguais ou melhores que o AM+BTLit. Da mesma forma que nos grupos de instâncias anteriores, neste o AM_RVND demonstra sua superioridade em relação as outras abordagens quando considerado o consumo de tempo. O AM_BT demonstra novamente, neste grupo ser capaz de encontrar soluções melhores que as outras abordagens.

Na Tabela 10, é apresentado um resumo dos resultados obtidos pelas abordagens testadas. São exibidos o número de melhores soluções encontradas dentre o total possível, a média do GAP e a média do tempo de CPU consumida pelas abordagens nos testes realizados. Ao analisar a diferença entre o GAP do AM_BT e AM_RVND é possível concluir que apesar de encontrar um número menor de melhores soluções, o AM_RVND encontra soluções próximas as melhores, sendo cerca de 6 vezes mais rápido que o AM_BT.

Tabela 9: Resultados computacionais para as instâncias do SET_4.

Instância	AM+BTLit			AM_BT				AM_RVND			
	V	Tam	T (s)	V	Tam	T (s)	GAP (%)	V	Tam	T (s)	GAP (%)
eil_51	6	443	3,9	6	**430**	7,2	-3,13	6	432	1,2	-2,50
berlin_52	7	**8586**	4,3	7	8641	3,4	0,65	7	8641	0,8	0,65
st_70	7	**721**	7,7	7	**721**	9,9	0,00	7	727	1,4	0,85
eil_76	6	557	10,9	6	**549**	19,4	-1,37	6	552	3,6	-0,74
pr_76	7	118386	9,8	7	**116596**	15,6	-1,51	7	116618	3,7	-1,49
kroa_100	6	22147	25,8	6	**22118**	25,1	-0,13	6	22223	4,0	0,34
kroc_100	6	21509	21,4	6	**20961**	26,7	-2,55	6	**20961**	4,3	-2,55
krod_100	6	21804	21	6	**21585**	40,3	-1,00	6	**21585**	4,0	-1,00
rd_100	6	8514	24,5	6	**8255**	29,9	-3,05	6	8486	5,4	-0,33
eil_101	6	667	21,7	6	**649**	46,5	-2,68	6	650	7,3	-2,61
ch_150	6	6729	59	6	**6604**	184,6	-1,86	6	6611	31,6	-1,75
tsp_225	8	4873	226,2	7	4705	966,9	-3,45	7	**4684**	71,3	-3,87
a_280	7	2920	409,7	6	2757	1631,9	-5,61	6	**2713**	171,5	-7,12
pcb_442	7	59670	2207,9	6	58682	4671,5	-1,66	7	**57514**	989,3	-3,61
pr_1002	7	308998	52028,5	7	304489	187197,0	-1,46	7	**303615**	43596,3	-1,74
Média			3672,2			12991,7	-1,92			2993,0	-1,83

Tabela 10: Resumo dos resultados.

	AM+BTLit	AM_BT	AM_RVND
Melhores soluções	42/131	108/131	86/131
Média GAP (%)	0,00	-1,22	-1,14
Média tempo (s)	1431,45	4609,02	770,23

Na próxima seção, serão expostas as considerações finais com base nos resultados obtidos pelas heurísticas propostas e sugestões de trabalhos futuros.

5. Conclusão

O PCVSH consiste em um problema de fácil entendimento, mas de difícil otimização e que pode ser aplicado em diversas situações práticas. Neste artigo, são propostas duas heurísticas baseadas em um algoritmo memético com operadores de vizinhança definidos na literatura.

As heurísticas propostas conseguem encontrar soluções melhores para a maioria das instâncias. A abordagem AM_RVND destaca-se por demandar um menor tempo de CPU em relação as outras abordagens, o que fornece uma boa relação entre a qualidade da solução e o tempo gasto. A heurística AM_BT é muito eficaz, mas precisa ser refinada com o intuito de diminuir o seu tempo de CPU.

Os resultados obtidos com as heurísticas propostas AM_BT e AM_RVND foram comparados com os resultados da heurística AM+BTLit, reimplementada segundo Castro *et al.*,

(2013). Após a análise dos resultados obtidos, concluiu-se que a heurísticas propostas são altamente competitivas quanto a qualidade das soluções. Em relação ao tempo de CPU o melhor é o AM_RVND. Para trabalhos futuros, será utilizada a ferramenta disponibilizada por López-Ibáñez et al., (2011) para calibrar os parâmetros utilizados nos algoritmos.

Sugere-se que novas abordagens heurísticas e variações do problema sejam abordadas. Pode-se considerar a inclusão de características como janela de tempo para os clientes e custo de visita aos hotéis. É possível adaptar a estratégia proposta para tratar outros problemas, como o Problema do Caixeiro Viajante Preto e Branco (BOURGEOIS *et al.*, 2001).

Agradecimentos

Os autores agradecem a CAPES pelo apoio financeiro à execução do trabalho.

Referências

Amorin, L. E., Gonçalves, L. B. e Magalhães, S. V. G., (2012), Um algoritmo memético para solução do problema de mínima latência, *Simp. Brasileiro de Pesquisa Operacional*, 2247-2258.
Applegate, D., Bixby, R., Chvátal, V. e Cook, W., Concorde TSP solver, (http://www.tsp.gatech.edu/concorde, 2006).
Bektas, T. (2006), The multiple traveling salesman problem: an overview of formulations na solution procedures, *Omega*, 34, 209-219.
Bourgeois, M., Laporte, G. e Semet, F. (2001), Heuristics for the black and White traveling salesman problem, *Computers and Operations Research*, 30, 1, 75-85.
Campos, S. C., Arroyo, J. E. C. e Gonçalves, L. B. (2013), Uma heurística GRASP-VND para o problema de sequenciamento de tarefas num ambiente assembly flowshop com três estágios e tempos de setup dependentes da sequência, *Simp. Bras. de Pesquisa Operacional*, 2147-2158.
Castro, M., Sorensen, K., Vansteenwegen, P. e Goos, P. (2012), A simple GRASP+VND for the TSPHS, Working paper 2012/24, Faculteit Toegepaste Economische Wetenschappen, Universiteit Antwerpen.
Castro, M., Sorensen, K., Vansteenwegen, P. e Goos, P. (2013), A memetic algorithm for the travelling salesperson problem with hotel selection, *C. and Operations Research*, 40, 1716-1728.
Dijkstra, E. W. (1959), A note on two problems in connexion with graphs, *Numerische Mathematik*, 1, 1, 269-271.
Garey, M. R. e Johnson, D. S., Computers and Intractability. A guide to the theory of NP-Completeness, *W. H. Freeman and Company*, 1979.
Glover, F. (1989), Tabu search – part I, *ORSA Journal on Computing*, 1, 3, 190-206.
Glover, F. (1990), Tabu search – part II, *ORSA Journal on Computing*, 2, 1, 4-32.
Hansen, P., Mladenović, N., Variable Neighborhood Search. In *Handbook of Metaheuristics*, 145-184, 2003.
Lin, S., (1965), Computer solutions of the traveling salesman problem, Bell System Technical Journal, 44, 10, 2245-2269.
Lin, S. e Kernighan, B. W. (1973), An effective heuristic algorithm for the travelling salesman problem, *Operations Research*, 21, 2, 498-516.
López-Ibáñez, M., Dubois-Lacoste, J., Stutzle, T. e Birattari, M., The irace package: Iterated racing for automatic algorithm configuration, Technical Report Series, Université Libre de Bruxelles, (http://iridia.ulb.ac.be/irace/ , 2011).
Merz, P., (2002), A comparison of memetic recombination operators for the traveling salesman problem, *Proc. of the Genetic and Evolutionary Computation Conference*, GECCO '02, 472-479.
Penna, P. H. V., Subramanian, A. e Ochi, L. S. (2013), Na iterated local search heuristic for the heterogeneous fleet vehicle routing problem, *Journal of Heuristics*, 19, 201-232.
Subramanian, A., Drummond, L. M. A., Bentes, C., Ochi, L. S. e Farias, R. (2010), A parallel heuristic for the vehicle routing problem with simultaneous pick-up and delivery, *Computers & Operations Research*, 37, 11, 1899-1911.
Talbi, E-G., Metaheuristics: from design to implementation, *Jonh Wiley and Sons Inc*, (2009).
Vansteenwegen, P., Souffriau, W. e Sorensen, K. (2012), The travelling salesperson problem with hotel selection, *Journal of the Operational Research Society*, 63, 207-217.