O PROBLEMA DO CAIXEIRO VIAJANTE COM PASSAGEIROS E LOTAÇÃO

Ranmsés E. M. Bastos
Universidade Federal do Rio Grande do Norte
Campus Universitário, Lagoa Nova, Natal, RN
ranmses@gmail.com

Marco C. Goldbarg
Universidade Federal do Rio Grande do Norte
Campus Universitário, Lagoa Nova, Natal, RN
marcocgold@gmail.com

Elizabeth F. G. Goldbarg
Universidade Federal do Rio Grande do Norte
Campus Universitário, Lagoa Nova, Natal, RN
beth@dimap.ufrn.br

RESUMO

O Problema do Caixeiro Viajante com Passageiros e Lotação é uma versão do Problema do Caixeiro Viajante clássico onde o caixeiro é o motorista de um veículo que compartilha os custos de viagem com passageiros. Além de dividir os custos do percurso, o caixeiro pode se valer, também, dos descontos das *high-occupancy vehicle lanes*, que são faixas de trânsito que isentam veículos lotados do pagamento de pedágio. Este artigo apresenta um modelo matemático resumido para este problema e um algoritmo baseado nas meta-heurísticas *Simulated Annealing* e Busca em Vizinhança Variável. Os resultados dos algoritmos heurísticos são comparados às soluções ótimas obtidas por um algoritmo exato.

PALAVRAS CHAVE. Ridesharing. Carpool. Simulated Annealing**.**

Tópicos (MH – Meta-heurísticas. OC - Otimização Combinatória. L&T - Logística e Transportes)

ABSTRACT

The Traveling Salesman with Passengers and High Occupancy Problem is a version of the classic Traveling Salesman Problem in which the salesman is the driver of a vehicle who shares travels' expenses with passengers. Besides shared expenses, the driver also benefits from discounts of high-occupancy vehicle lanes, i.e. traffic lanes in which high occupancy vehicles are exempted from tolls. This paper presents a succinct mathematical model for this problem and algorithms based on Simulated Annealing and Variable Neighborhood Search metaheuristics. The results of the heuristic algorithms are compared with the optimal solutions obtained by an exact algorithm.

KEYWORDS. Ridesharing. Carpool. Simulated Annealing.

Paper topics (MH – Metaheuristics. OC - Combinatorial Optimization. L&T - Logistics and Transport)

1. Introdução

Este trabalho apresenta um novo problema de roteamento intitulado Problema do Caixeiro Viajante com Passageiros e Lotação (PCV-PL). O modelo é uma variante do Problema do Caixeiro Viajante (PCV) e incorpora elementos das classes *Pickup and Delivery* (P&D) e *Ridesharing*. O problema em questão inclui, no PCV tradicional, a consideração do valor econômico relativo ao aproveitamento do potencial de transporte solidário do veículo do caixeiro para reduzir as despesas do motorista e proporcionar um meio de transporte mais eficiente para os passageiros e de menor impacto sobre o meio ambiente. O problema está correlacionado com o Problema do Caixeiro Viajante com Passageiros (PCV-Pa) apresentado por Calheiros [2015], sendo dele uma variante. No PCV-Pa, um motorista deve percorrer um ciclo hamiltoniano sobre um conjunto de localidades e pode transportar passageiros com o objetivo de, com eles, ratear custos de viagem. No PCV-PL o caixeiro pode beneficiar-se adicionalmente pela isenção de pedágio em vias que possuam tal desconto. A exigência para a isenção é de que o veículo tenha uma lotação de passageiros maior ou igual a um certo valor mínimo. Essas vias são denominadas usualmente de *high-occupancy vehicle lanes* ou faixas *hov*. Em outras palavras, a parcela de custos referente ao pedágio é nula caso o veículo esteja suficientemente ocupado ao trafegar pelas conexões *hov*. Estas faixas especiais de trânsito estão presentes em várias partes do mundo e fazem parte de esforços governamentais para redução nominal do tráfego em rodovias. Além do incentivo financeiro da isenção de pedágio, é comum que as faixas sejam exclusivas e apresentem menor incidência de congestionamentos, bem como permitam evitar as filas ou afunilamentos nas praças de pedágio.

O PCV-PL está relacionado a outros problemas clássicos de roteamento. O PCV é um caso particular do PCV-PL quando não são oferecidos assentos para passageiros. Ao permitir o compartilhamento de assentos, o problema também pode ser incluído genericamente na classe dos problemas de *ridesharing* a qual é definida sob diferentes pontos de vista conforme Amey et al. [2011]. De uma forma geral, um problema da classe *ridesharing* corresponde à reunião de duas ou mais pessoas para partilhar uma única viagem em um veículo, sem levar em conta um acordo prévio ou um histórico de cooperação [Dailey et al. 1999]. Uma visão estruturada do problema pode ser obtida em [Furuhata et al. 2013].

Como um problema que embarca e desembarca passageiros ao longo da rota, o PCV-PL compartilha elementos com os problemas de roteamento com P&D. Parragh et al. [2008a, 2008b] revisam a literatura do P&D. O PCV-PL distingue-se do P&D tanto pela função objetivo que associa o ganho do problema à razão entre o custo de trafegar na via e o carregamento do veículo, quanto em virtude da programação de embarque e desembarque atender exclusivamente ao interesse do caixeiro. No PCV-PL não existe demanda de transporte a ser atendida. Uma das variantes do P&D que se aproxima do PCV-PL é o *Single Vehicle Routing Problem with Deliveries and Selective Pickups* [Gribkovskaiaa et al. 2008], que permite não satisfazer as coletas demandadas.

O PCV-PL ainda compartilha elementos com os problemas de roteamento com coleta, como *Travelling Salesman with profits* [Feillet et al. 2005], *Prize Collecting Travelling Salesman* [Balas 2004], *Attractive Travelling Salesman* [Erdoğan, et al. 2010], *Selective Travelling Salesman* [Gendreau et al. 1998] e *Orienteering Problems* [Vansteenwegen et al. 2011], contudo o ganho associado ao embarque do passageiro não é fixo ou limitado, sendo variável em função da rota adotada.

Poucos trabalhos que abordam a otimização de rotas em função de ganhos obtidos com ligações do tipo *hov* são relatados na literatura. Wang et al. [2013] abordam uma variante P&D com ligações *hov*, janelas de tempo e *ridesharing*.

Neste trabalho é apresentado um algoritmo exato e algoritmos baseados nas meta-heurísticas *Simulated Annealing* e Busca em Vizinhança Variável para o PCV-PL. São reportados os resultados de um experimento computacional onde foram comparadas as abordagens propostas.

As próximas seções estão organizadas da seguinte forma: a seção 2 apresenta o PCV-PL; a seção 3 apresenta os algoritmos desenvolvidos; a seção 4 detalha os experimentos computacionais e a seção 5 apresenta a conclusão do trabalho.

2. Problema do Caixeiro Viajante com Passageiros e Lotação

Mesclando a classe dos problemas de *ridesharing* com a classe dos problemas do caixeiro viajante, o problema pode ser entendido como uma variante do Caixeiro Viajante Capacitado em que o caixeiro pode reduzir seus custos através de compartilhamento de despesas de viagem com eventuais passageiros embarcados e, adicionalmente, beneficia-se de incentivos sociais pelo transporte solidário. No PCV-PL considera-se que a disposição das localidades e ligações rodoviárias associadas ao problema são representadas por um grafo ponderado $G = (N, M)$ onde $N = \{1,...,n\}$ modela o conjunto de localidades e $M = \{1,...,m\}$ o conjunto de ligações rodoviárias existentes entre as localidades. O ciclo do caixeiro é iniciado na localidade de número 1.

Os passageiros desejam, de uma forma geral, embarcar e desembarcar em localidades que pertencem ao ciclo do caixeiro. Consequentemente, os passageiros podem compartilhar assentos no veículo do caixeiro em parte de sua rota, desde que a cidade de embarque do passageiro venha antes da cidade de desembarque. O caixeiro não cobra tarifa fixa pelos deslocamentos e sim rateia as despesas do deslocamento de cada trecho com os passageiros embarcados no trecho. Neste trabalho considera-se que os passageiros não são sensíveis ao caminho percorrido entre a cidade de embarque e a cidade destino, ou seja, ao número de cidades intermediárias entre o embarque e o desembarque. No caso examinado neste trabalho considera-se que existe apenas 1 pessoa solicitando transporte em cada cidade, i.e., o número total de solicitações é n. Cada passageiro possui um limite de orçamento, ou seja, um valor máximo que o mesmo está disposto a pagar por suas despesas de deslocamento. A despesa total do deslocamento não deve ultrapassar t_l unidades monetárias, $l = 1,...,n$. O l-ésimo passageiro deseja iniciar seu caminho na cidade P_l e o terminá-lo na cidade Q_l, $Q_l \neq P_l$, $l = 1,...,n$. No caso particular examinado, $Q_1 \neq 1$.

Ao embarcar na cidade P_l, o passageiro l passa a dividir em rateio uniforme, com os demais ocupantes do veículo, outros passageiros e o motorista, as despesas dos trechos em que l seguir embarcado. A parte da despesa que cabe a cada ocupante, passageiros e motorista, para trafegar em cada trecho (i, j), $(i, j) \in M$, é obtida pela divisão do custo do deslocamento entre as cidades i e j pelo número de pessoas que ocupam o veículo na aresta (i, j).

Quando o veículo do caixeiro atravessar uma aresta que está habilitada ao benefício do desconto associado ao transporte solidário, chamada de aresta *hov*, e o veículo possuir a lotação mínima exigida para habilitar o incentivo, o valor da aresta é decrescido do valor do desconto. Consequentemente, o valor de rateio da aresta *hov* sofre um decréscimo em seu custo operacional caso o veículo por ela circule atendendo a lotação socialmente incentivada. De uma forma geral, trafegar com a lotação exigida em arestas *hov* sugere a possibilidade de reduções adicionais e que podem ser somadas às reduções obtidas pelo rateio.

A solução do PCV-PL consiste na determinação de um ciclo Hamiltoniano no grafo G e da lista de pessoas, passageiros, que terão suas solicitações de transporte atendidas. As restrições devem garantir que as condições de embarque e desembarque, os limites de orçamento dos passageiros e a capacidade do veículo sejam atendidas. As arestas do tipo *hov* devem ser diferenciadas em valor das arestas normais do problema e é necessário que restrições controlem a ativação do desconto de ocupação máxima. O conjunto *Hov* contém as arestas do tipo *hov*.

O objetivo é minimizar a função (1) onde d_{ij} denota o custo da aresta (i, j); f_{ij} é uma variável Booleana igual a 1 se a aresta (i, j) pertence ao ciclo Hamiltoniano definido para o caixeiro e 0 caso contrário; c_{ij} é a redução prevista para o custo da aresta (i, j) caso o carro circule em (i, j) com lotação que garanta o desconto na via; w_{ij} é uma variável Booleana igual a 1 se o caixeiro atravessa a aresta (i, j) com lotação que garanta o desconto na via e 0 caso contrário; L denota o conjunto de pessoas demandando transporte e v_{ij}^{l} é uma variável Booleana que possui valor 1 se o carona l foi transportado na aresta (i, j) e 0 em caso contrário.

$$\sum_{i,j \in N} \frac{d_{ij} f_{ij} - c_{ij} w_{ij}}{\sum_{l \in L} v_{ij}^{l} + 1} \qquad (1)$$

A função (1) é de natureza não linear. Um modelo matemático deve forçar que o produto $c_{ij}w_{ij}$ seja igual a zero se $(i, j) \notin$ Hov. Todavia, para que seja diferente de zero, é necessário que $(i, j) \in$ Hov e a lotação do veículo expressa por $\sum_{l \in L} v_{ij}^{l}$ seja maior ou igual a h_{ij}, a lotação mínima do veículo exigida na aresta (i, j) para que o desconto seja alcançado. Assim, a variável w_{ij} deve receber valor 1 quando $\sum_{l \in L} v_{ij}^{l} \geq h_{ij}$.

O total de passageiros transportados em uma aresta (i, j) não deve ultrapassar a capacidade R do veículo. Esta restrição está expressa em (2).

$$R f_{ij} - \sum_{l \in L} v_{ij}^{l} \geq 0 \qquad \forall (i,j) \in M \qquad (2)$$

A condição de limite de orçamento de cada passageiro pode ser expressa como em (3).

$$\sum_{i \in N} \sum_{j \in N} \frac{d_{ij} v_{ij}^{l}}{\sum_{k \in L} v_{ij}^{k} + 1} - t_l \leq 0 \qquad \forall l \in L \qquad (3)$$

Como uma generalização do PCV, a solução do PCV-PL é pelo menos tão difícil quanto a solução do PCV, um conhecido problema NP-Árduo [Garey e Jonhson 1979]. Todavia, o número de possíveis decisões no PCV-PL é maior do que no PCV, uma vez que o custo do ciclo não pode ser obtido *a priori* através da soma do custo das arestas por depender do carregamento do veículo. Por sua vez, o carregamento depende da sequência de visitas, da decisão de embarque e da capacidade do veículo. O esquema de embarques e desembarques ótimo calculado sobre a rota ótima do caixeiro viajante clássico pode não ser a solução ótima para o PCV-PL, uma vez que o carregamento ótimo do veículo dependerá também do destino desejado pelos passageiros. Há que se determinar uma rota de baixo custo e que permita um carregamento excelente do veículo. A dificuldade da solução desse problema é sugerida na própria modelagem simplificada, pois se evidencia a necessidade de alcançar a solução de um problema de programação inteira com função objetivo não linear e pelo menos uma restrição não linear de tarifa, além de um conjunto de restrições lógicas que garantam o comportamento em função do desconto da lotação máxima.

3. Algoritmos para o PCV-PL

Foi desenvolvido um algoritmo exato do tipo *backtracking* de forma a estabelecer um referencial de comparação de desempenho para os algoritmos heurísticos para as instâncias onde foi possível encontrar soluções ótimas. Uma solução s para o problema é composta de duas partes: uma permutação de *n* vértices que define a rota do caixeiro e uma lista de atribuição de passageiros. A lista com a permutação dos *n* vértices será notada pelo próprio s e a lista de atribuição de passageiros será notada por L.

3.1. Algoritmo Exato

O Algoritmo 1 descreve, em pseudocódigo, a estratégia utilizada para determinação da solução exata. No passo 1 são gerados todos os ciclos Hamiltonianos com início e fim no vértice de origem. O melhor custo de uma solução é iniciado no passo 2. No laço dos passos 3-6 do algoritmo 1 é chamado o procedimento $Embarque$ que define quais passageiros serão embarcados. No passo 3, é feita uma verificação para evitar que soluções não promissoras sejam desenvolvidas.

O teste é feito verificando se a melhor solução gerada até o momento tem custo maior que a solução parcial sendo gerada. O limite inferior é definido pela divisão entre o somatório dos custos do ciclo supondo que todos os descontos de pedágios são ativados ($s.comprimento$) e a capacidade R do veículo. Uma árvore de busca pelo melhor esquema de carregamento de s é iniciada no passo 4. O custo inicial da solução é dado pelo custo da rota do caixeiro em s. No procedimento $Embarque$ é construída a árvore de busca para alocação de passageiros em s. O custo do nó da árvore é atualizado sempre que uma solução é criada ou modificada (passos 1, 6 e 9 do procedimento $Embarque$). Para cada cidade $s[i]$ é verificado se o candidato a embarque naquela cidade, p, pode ser embarcado. Caso seja possível, é criado um nó na árvore de busca considerando a inclusão deste passageiro, ou seja, $L[s[i]]$ recebe valor 1 e o custo com a inclusão de p é calculado (passos 4 e 5 do procedimento $Embarque$). Também, em toda iteração, é criado um nó na árvore de busca onde o passageiro p não é embarcado (passos 7 e 8 do procedimento $Embarque$).

Algoritmo 1 – *BT*	
1	**Gere** todos os ciclos hamiltonianos e coloque no conjunto *CH*
2	$melhor.custo \leftarrow infinito$
3	**Para cada** $s \in CH$, **se** $melhor.custo > (s.comprimento)/R$
4	**Inicie** uma árvore de busca β tal que $\beta.raiz \leftarrow s$
5	$\beta.raiz.custo \leftarrow calcula\_custo(s)$
6	**Execute** $Embarque(\beta.raiz, L)$
Procedimento *Embarque*$(node, L)$	
1	**Se** $node.custo \leq melhor.custo$, **faça** $melhor.custo \leftarrow node.custo$
2	**Para** $i \leftarrow 1$ até n
3	**Se** o embarque do passageiro em $s[i]$ é viável **faça**
4	$L[s[i]] \leftarrow 1$; $node.right.custo \leftarrow calcula\_custo(s)$
5	$Embarque(node.right, L)$
6	**Se** $node.right.custo \leq melhor.custo$ **faça** $melhor.custo \leftarrow node.right.custo$
7	$L[s[i]] \leftarrow 0$
8	$Embarque(node.left, L)$
9	**Se** $node.left.custo \leq melhor.custo$ **faça** $melhor.custo \leftarrow node.left.custo$

Algoritmo 1 – *Backtracking*

3.2. Algoritmos Heurísticos

Os algoritmos heurísticos desenvolvidos foram inspirados nas meta-heurísticas *Simulated Annealing* (SA) e Busca em Vizinhança Variável (VNS). Tal escolha levou em conta os resultados promissores relatados para o SA em problemas de diversas áreas, bem como a flexibilidade que a estratégia VNS permite para buscas locais. A composição das duas abordagens supostamente tem potencial para uma evolução progressiva da busca e fuga de ótimos locais através do esquema *annealing*, conhecido como critério de Metropolis, em referência ao trabalho tido como precursor do desenvolvimento do método [Metropolis et al. 1953]. Por outro lado, herda a capacidade de exploração sistemática e aprofundada de boas vizinhanças da proposta VNS.

Basicamente, o SA pode partir de qualquer solução inicial s. Seu procedimento iterativo consiste na geração de uma solução vizinha $v \in \varphi(s)$, onde $\varphi(s)$ representa a vizinhança de s, e no teste de aceitação para a transição de s para v. A aceitação da transição é realizada pela avaliação da variação da energia $\Delta = \xi(v) - \xi(s)$ entre as duas soluções. A probabilidade da aceitação da

transição $P_i(s \rightarrow v)$ é calculada pelo conjunto de equações (4). O parâmetro T_i, associado à probabilidade P_i no passo i, permite modular a probabilidade da aceitação da transição tanto em função da diferença de energia entre as soluções quanto do estágio i da busca.

$$P_i(s \rightarrow v) = \begin{cases} e^{-\Delta/T_i}, & se\ \Delta \geq 0 \\ 1, & se\ \Delta < 0 \end{cases} \quad (4)$$

O termo $e^{-\Delta/T_i}$ representa a probabilidade da busca evoluir para um estado pior (mais energético) que o estado i, o que diminui a chance de estagnação da busca em ótimos locais. A cada rodada o sistema é lentamente resfriado e o *loop* descrito se prolonga até que seja alcançada uma temperatura mínima T_{min} ou até que sejam analisados *It* estados (equivalente à quantidade de iterações). Para cada implementação é preciso definir os seguintes parâmetros: T_{min} e *It*, já descritos; T_0 - *temperatura inicial*; T_i - *mecanismo de decréscimo da temperatura*; φ - *função geradora da vizinhança*; e ξ - *função energia*. No presente trabalho foi adotado o mecanismo homogêneo para T_i, o qual encontra-se descrito na equação (5).

$$T_{i+1} = \alpha . T_i \qquad 0 < \alpha < 1 \quad (5)$$

A variação da vizinhança foi inserida no SA como um esquema de geração da solução $v \in \varphi(s)$. A função vizinhança $\varphi(s)$ passa a ser uma interface para uma escolha equiprovável de k estruturas distintas de vizinhança $\varphi_1(s), \varphi_2(s), \ldots, \varphi_k(s)$. Foram adotadas $k = 4$ estruturas de vizinhança para variação da rota, cada uma delas baseada nos operadores: *troca, inserção, inversão e vizinho mais próximo*. No operador *troca*, troca-se a posição de duas cidades no ciclo. Por exemplo, na figura 1, são trocadas de posição as cidades 4 e 2. No operador *inserção* uma cidade é removida de sua posição e inserida após outra. Por exemplo, na figura 1 a cidade 4 é removida e inserida após a 9. No operador *inversão*, uma sequência de cidades é invertida. Por exemplo, a sequência entre as cidades 4 e 2 é invertida. No operador *vizinho mais próximo*, é escolhida uma cidade e é colocada ao lado da primeira sua cidade mais próxima. Por exemplo, na Figura 1 a cidade 2 é colocada ao lado da cidade 4.

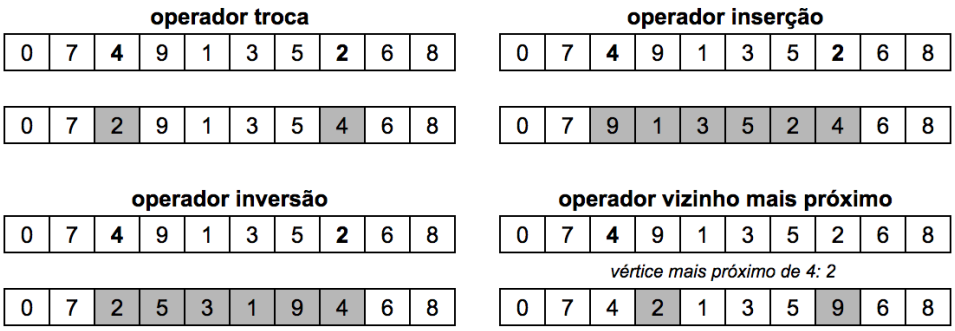

Figura 1 – Operadores para variação da rota

Para atribuir passageiros à rota, foi adotada a estratégia construtiva descrita no Algoritmo 2, designado por *Heurística de Carregamento (HC)*. A heurística *HC* recebe como parâmetros de entrada uma solução s ainda sem passageiros atribuídos e a lista *Tarmax* que contém o máximo que cada passageiro está disposto a pagar. A saída de *HC* é a rota s e a atribuição de passageiros na lista L. No passo 1 do Algoritmo 2, a variável θ recebe valor falso. Esta variável receberá valor verdadeiro quando for atribuído um carregamento de passageiros válido para a rota da solução s. Inicialmente, todos os candidatos a embarque estão desmarcados e somente o motorista está no carro (passo 2 do Algoritmo 2). O laço entre as linhas 3 e 16 faz a alocação de passageiros. No passo 5 são verificadas as primeiras condições de embarque do passageiro. Tais condições são:

a) o ponto de embarque do passageiro encontra-se antes do seu ponto de desembarque na rota e
b) existe possibilidade de a tarifa final do passageiro ser menor que sua tarifa limite, ou seja, se a tarifa limite do passageiro é maior que a divisão entre o somatório dos custos dos trechos entre o embarque e o desembarque, supondo que todos os descontos de pedágios são ativados, e a capacidade R.

O embarque do passageiro somente é aceito se o passageiro não tiver sido marcado em uma iteração anterior e se ainda existir assento disponível, conforme verificado no passo 6 do Algoritmo 2. O custo da solução (rota e embarques) é calculado no passo 7. As linhas 8-14 verificam se existe algum passageiro excedendo a tarifa máxima admitida para ser marcado. No caso de existir mais de um, é marcado o passageiro que tiver a maior diferença entre o seu limite e a tarifa atual. O passageiro marcado não é considerado em futuras iterações do algoritmo. Se o esquema de embarque possui algum passageiro na condição descrita, então o esquema é descartado e o algoritmo reinicia a construção da alocação de passageiros (linhas 15-16). Se o esquema é válido, o algoritmo é finalizado.

Algoritmo 2 – *HC*(*s*,*Tarmax*)	
1	$\theta \leftarrow FALSE$
2	**Desmarque** todos os passageiros; **Para** $i \leftarrow 1$ até n **faça** $L[i] \leftarrow 0$
3	**Enquanto** $\theta = FALSE$
4	**Para** $i \leftarrow 1$ até n
5	**Se** o embarque do passageiro p no vértice $s[i]$ é viável **faça**
6	**Se** p é desmarcado e existem assentos disponíveis **faça** $L[s[i]] \leftarrow 1$
7	$s.custo \leftarrow calcula\_custo(s)$
8	$maxtar \leftarrow 0; j \leftarrow 0$
9	**Para** $i \leftarrow 1$ até n
10	**Se** $L[i] = 1$ **faça** $tarifa[i] \leftarrow calcula\_tarifa(i)$
11	**Se** $maxtar < tarifa[i] - Tarmax[i]$ **faça**
12	$maxtar \leftarrow tarifa[i] - Tarmax[i]; j \leftarrow i$
13	**Se** $j \neq 0$ **faça**
14	**Marque** o passageiro j
15	**Para** $i \leftarrow 1$ até n **faça** $L[i] \leftarrow 0$
16	**Caso contrário,** $\theta \leftarrow TRUE$

Algoritmo 2 – Heurística de Carregamento

Foram desenvolvidos dois algoritmos heurísticos que possuem o mesmo arcabouço: *Simulated Annealing com operadores* (*SAop)* e *Simulated Annealing com Busca Local (SABL)*. Os dois distinguem-se pela aplicação ou não de buscas locais na geração de vizinhos. O método para gerar a solução inicial é o mesmo em ambos os casos e consiste em gerar uma rota aleatória, aplicar a heurística de Lin e Kernighan [1973], *LKH*, e atribuir os embarques conforme o Algoritmo 2.

Partindo da solução s, o Algoritmo 3 gera um vizinho para s por meio da aplicação única de um operador selecionado em sorteio aleatório equiprovável (passo 6), de modo que a cada iteração é avaliada apenas uma nova solução. O carregamento de passageiros na nova rota é realizado com o Algoritmo 2 (passo 7 do Algoritmo 3). A nova solução, v, é avaliada. O valor de avaliação associado a uma solução é notado por $\xi(\,)$. A avaliação é feita pela expressão (1).

Algoritmo 3 – *SAop*(*It*,*T*,*Tarmax*)	
1	**Crie** a rota da solução inicial (s)
2	$s \leftarrow HC(s,Tarmax)$; $T \leftarrow T_0$; $melhor \leftarrow s$
3	**Enquanto** ($T \geq T_{min}$)
4	**Para** $i \leftarrow 1$ até It
5	**Sorteie** um operador γ
6	**Aplique** γ à rota de s gerando s'
7	$v \leftarrow HC(s',Tarmax)$
8	$\Delta \leftarrow \xi(v) - \xi(s)$
9	**Se** $\Delta < 0$ **faça**
10	$s \leftarrow v$
11	**Se** $\xi(melhor) > \xi(s)$ **faça** $melhor \leftarrow s$
12	**Se** $\Delta \geq 0$ **faça** v é aceita para substituir s com probabilidade $e^{-\Delta/T}$
13	**Atualize** T

Algoritmo 3 – *Simulated Annealing com operadores*

Algoritmo 4 – *SABL*(*It*,*T*,*Tarmax*)	
1	**Crie** a rota da solução inicial (s)
2	$s \leftarrow HC(s,Tarmax)$; **Aplique** *Busca Local para Carregamento* em s
3	$T \leftarrow T_0$; $melhor \leftarrow s$
4	**Enquanto** ($T \geq T_{min}$)
5	**Para** $i \leftarrow 1$ até It
6	**Sorteie** um operador γ
7	**Execute** *Busca Local para Rota* em s utilizando γ gerando v
8	$\Delta \leftarrow \xi(v) - \xi(s)$
9	**Se** $\Delta < 0$ **faça**
10	$s \leftarrow v$
11	**Se** $\xi(\text{melhor}) > \xi(s)$ **faça** $melhor \leftarrow s$
12	**Se** $\Delta \geq 0$ **faça** v é aceita para substituir s com probabilidade $e^{-\Delta/T}$
13	**Se** v substituiu s **faça** /* Neste caso $s \leftarrow v$ em passo anterior */
14	*Busca Local para Carregamento* em s gerando v'
15	**Se** $\xi(v') < \xi(s)$ **faça** $s \leftarrow v'$
16	**Se** $\xi(melhor) > \xi(s)$ **faça** $melhor \leftarrow s$
17	**Atualize** T

Algoritmo 4 – *Simulated Annealing com Buscas Locais*

O Algoritmo 4 faz uso de buscas locais para rota e carregamento. A inserção de tal artifício objetivou explorar o espaço de soluções de maneira mais completa que o Algoritmo 3, abrangendo regiões do espaço que potencialmente não foram por este visitadas.

A *Busca Local para Carregamento* (passos 2 e 14 do Algoritmo 4) mantém inalterada a rota e objetiva encontrar um esquema de carregamento que resulte em uma solução de menor custo. São avaliados todos os passageiros embarcáveis que não embarcaram na solução original. O passageiro que possui o trecho mais longo é inserido no esquema de embarque. A nova solução é avaliada. Se ela for viável, é tomada como resultado uma vez que é, necessariamente, melhor que a solução prévia. Caso contrário, é iniciado o procedimento de busca por uma solução viável que obrigatoriamente inclua o passageiro com o maior trecho e exclua o embarque de, no máximo, 2 passageiros que estão na solução corrente. Seguindo este critério, todas as combinações de 1 ou 2 são então avaliadas e a saída do procedimento é a melhor solução dentre as avaliadas na busca e a própria solução inicial.

A *Busca Local para Rota* (passo 7 do Algoritmo 4) altera a rota e consequentemente o esquema de carregamento. O processo utiliza um dos operadores de modificação de rota, selecionado em sorteio equiprovável. O operador é aplicado sistematicamente a cada um dos vértices da rota, gerando um conjunto de novas rotas com cardinalidade $O(n)$. Cada rota do conjunto é então carregada por meio do Algoritmo 2 e as novas soluções são avaliadas. A saída é a melhor solução dentre aquelas pertencentes ao conjunto gerado, que é obrigatoriamente diferente da solução original.

4. Experimento Computacional

Os algoritmos foram implementados na linguagem *C++*. As execuções dos algoritmos *SAop* e *SABL* se deram num computador portátil com memória RAM de 8 GB, processador *Intel Core i7* modelo *3615QM* com 2,3 GHz e sistema operacional *OSX 10.11.4*. A determinação das soluções exatas pelo algoritmo BT foi realizada numa estação de trabalho com memória RAM de 96 GB, 2 processadores *Intel Xeon* modelo *X5690* com 3,47 GHz e sistema operacional *CentOS 6.7*.

Para avaliar o desempenho dos algoritmos, foi elaborado um conjunto de casos de teste composto por grafos simétricos com custos das arestas variando aleatoriamente entre 50 e 500. Em cada vértice existe um passageiro cujo destino é selecionado em sorteio aleatório com distribuição uniforme. A tarifa limite para os passageiros é calculada em função de um percentual da árvore geradora mínima do grafo, variando entre 25% e 50%, conforme o caso teste. A designação das arestas *hov* foi realizada em sorteio aleatório equiprovável sobre as arestas da instância, com base em um percentual do total de arestas previamente fixado *S*, $S \in \{10\%; 20\%; 25\%\}$. Os descontos que formam a variável c_{ij} foram designados em sorteio aleatório equiprovável sobre as arestas *hov* em um percentual previamente fixado *F*, $F \in \{20\%; 25\%; 30\%; 40\%\}$. O desconto é ativado quando, ao percorrer uma aresta *hov*, o veículo encontra-se pleno em sua capacidade $R \in \{3; 4\}$. Foram criadas 12 classes de instâncias, cada classe possuindo 25 (vinte e cinco) problemas e sendo caracterizada por um par dos valores permitidos para *S* e *F*. Cada instância é unicamente identificada pelo par *Cl* - classe - e *In* - dígito identificador de instância relacionado ao valor *N,* o número de vértices da instância. Para a realização do experimento, foram considerados 5 problemas de cada classe dos casos teste com $N = 10$, totalizando 60 instâncias. As instâncias estão disponíveis em <http://www.dimap.ufrn.br/lae/projetos/CaRS.php>. O experimento consistiu na execução de 100 execuções de cada algoritmo heurístico para cada instância não utilizada no procedimento de ajustes de parâmetro.

Tabela 1 – Parâmetros dos algoritmos *SAop* e *SABL*.

Parâmetro	Valores possíveis	*SAop*	*SABL*
T_0	{1; 20; 40; 60; 80; 100}	1	60
T_{min}	{0,01; 0,001; 0,0001; 0,00001}	0,00001	0,001
α	[0,75; 0,99]	0,84	0,78
It	{100; 500; 1000; 1500; 2000}	2000	2000

O ajuste de parâmetros foi realizado por meio da ferramenta IRACE - *Iterated Racing for Automatic Algorithm Configuration* [López-Ibáñez et al. 2011], que executa um processo automatizado de afinamento. Foram selecionadas de forma aleatória, do banco de 60 instâncias, 10 para o conjunto de treinamento. As configurações selecionadas estão na Tabela 1.

Tabela 2 – Resultados do experimento computacional

	BT		*SAop*					*SABL*				
Inst.	**Exata**	**T (s)**	**Mín.**	**Méd.**	**σ**	**Δ%**	**Tm (s)**	**Mín.**	**Méd.**	**σ**	**Δ%**	**Tm (s)**
01-01	590,83	127,07	590,83	620,56	27,49	0,000	0,15	590,83	591,93	6,29	0,000	0,57
01-02	661,33	119,03	661,33	667,66	21,50	0,000	0,15	661,33	661,33	0,00	0,000	0,58
01-03	567,93	115,68	567,93	640,38	79,72	0,000	0,15	567,93	569,27	13,34	0,000	0,60
01-04	837,67	120,67	837,67	851,58	19,49	0,000	0,15	837,67	837,67	0,00	0,000	0,58
02-01	582,17	153,40	582,17	626,77	50,66	0,000	0,15	582,17	582,89	7,20	0,000	0,61
02-02	868,50	217,99	868,50	915,87	46,31	0,000	0,15	868,50	869,59	7,37	0,000	0,64
02-03	653,83	109,52	653,83	694,06	38,64	0,000	0,15	653,83	653,83	0,00	0,000	0,59
02-04	883,67	89,69	883,67	897,19	28,34	0,000	0,16	883,67	883,67	0,00	0,000	0,71
03-01	394,73	251,09	394,73	404,32	21,28	0,000	0,16	394,73	394,73	0,00	0,000	0,74
03-02	548,42	199,37	548,42	659,23	79,18	0,000	0,16	548,42	555,54	28,35	0,000	0,67
03-03	469,00	189,82	469,00	487,96	24,67	0,000	0,17	469,00	469,00	0,00	0,000	0,76
03-04	608,05	261,16	608,47	629,47	48,81	**0,069**	0,17	608,05	608,05	0,00	0,000	0,81
03-05	579,17	176,76	579,17	580,43	8,58	0,000	0,16	579,17	579,17	0,00	0,000	0,66
04-01	443,42	122,51	443,42	461,04	27,47	0,000	0,15	443,42	443,42	0,00	0,000	0,62
04-02	386,13	149,79	386,13	488,27	78,06	0,000	0,16	386,13	401,79	44,77	0,000	0,68
04-03	389,75	175,35	389,75	465,20	64,33	0,000	0,17	389,75	395,57	17,61	0,000	0,78
04-04	333,25	150,15	354,75	401,17	85,90	**6,452**	0,16	333,25	333,25	0,00	0,000	0,70
04-05	692,40	140,47	692,40	769,95	91,67	0,000	0,16	692,40	692,40	0,00	0,000	0,71
05-01	454,67	128,24	454,67	457,53	16,43	0,000	0,15	454,67	454,67	0,00	0,000	0,57
05-03	695,42	221,45	695,42	718,37	39,17	0,000	0,15	695,42	695,81	1,47	0,000	0,57
05-04	521,92	270,04	557,08	572,41	20,62	**6,738**	0,15	521,92	521,92	0,00	0,000	0,60
05-05	585,67	232,58	585,67	586,90	7,04	0,000	0,15	585,67	585,67	0,00	0,000	0,62
06-02	636,75	215,24	641,92	669,32	26,33	**0,811**	0,15	641,92	642,69	4,40	**0,811**	0,60
06-04	462,67	153,14	462,67	478,55	28,01	0,000	0,16	462,67	462,67	0,00	0,000	0,67
06-05	441,58	162,53	453,33	488,03	42,79	**2,661**	0,16	441,58	443,23	4,10	0,000	0,63
07-01	309,33	277,01	309,33	332,14	28,42	0,000	0,15	309,33	309,33	0,00	0,000	0,66
07-04	510,39	363,35	510,39	514,85	13,95	0,000	0,15	510,39	510,39	0,00	0,000	0,64
07-05	565,88	548,27	567,63	575,93	11,69	**0,311**	0,14	565,88	565,88	0,00	0,000	0,62
08-01	465,60	591,39	465,60	482,70	22,89	0,000	0,15	465,60	465,72	1,16	0,000	0,72
08-02	314,96	315,67	314,96	324,22	13,63	0,000	0,17	314,96	315,87	3,00	0,000	0,80
08-03	452,86	392,94	452,86	460,75	13,94	0,000	0,15	452,86	452,99	0,65	0,000	0,65
08-05	386,37	351,80	386,37	390,24	8,40	0,000	0,15	386,37	386,37	0,00	0,000	0,70
09-01	445,03	362,05	445,03	459,79	14,81	0,000	0,16	445,03	445,03	0,00	0,000	0,67
09-02	451,40	319,41	472,73	495,07	19,47	**4,726**	0,15	451,40	455,45	8,41	0,000	0,64
09-03	420,20	452,86	420,20	431,79	14,45	0,000	0,15	420,20	420,32	0,91	0,000	0,61
09-04	489,93	378,41	489,93	500,42	15,04	0,000	0,14	489,93	489,93	0,00	0,000	0,56
09-05	350,17	327,36	350,17	352,83	9,41	0,000	0,15	350,17	350,17	0,00	0,000	0,59
10-01	504,43	401,92	508,60	525,16	18,32	**0,826**	0,14	504,43	504,43	0,00	0,000	0,55
10-02	429,23	317,58	429,23	472,15	33,94	0,000	0,14	429,23	429,66	2,99	0,000	0,53
10-03	608,17	418,62	608,17	620,00	16,53	0,000	0,14	608,17	608,17	0,00	0,000	0,50
10-05	427,30	389,76	427,30	430,51	8,32	0,000	0,15	427,30	427,30	0,00	0,000	0,57
11-01	466,42	643,42	466,42	485,76	21,22	0,000	0,15	466,42	466,45	0,36	0,000	0,61
11-03	417,22	1084,24	417,22	434,21	13,06	0,000	0,15	417,22	417,27	0,28	0,000	0,65
11-04	456,70	1311,22	456,70	473,97	20,11	0,000	0,15	456,70	456,70	0,00	0,000	0,66
11-05	458,28	749,36	458,28	469,21	11,36	0,000	0,14	458,28	458,28	0,00	0,000	0,62
12-01	428,07	919,52	428,07	444,28	15,43	0,000	0,15	428,07	428,07	0,00	0,000	0,64
12-02	437,07	722,92	437,07	447,57	15,12	0,000	0,15	437,07	437,07	0,00	0,000	0,61
12-03	504,10	1028,79	504,10	511,91	8,46	0,000	0,14	504,10	504,10	0,00	0,000	0,60
12-04	414,78	643,84	414,78	422,63	9,75	0,000	0,14	414,78	414,78	0,00	0,000	0,61
12-05	434,60	754,31	441,52	446,70	13,31	**1,592**	0,14	434,60	434,60	0,00	0,000	0,62

A Tabela 2 apresenta os resultados do algoritmo exato *BT* e dos algoritmos heurísticos *SAop* e *SABL* para as instâncias do banco excetuando-se aquelas empregadas no passo de calibração de parâmetros, totalizando 50 casos de teste. Cada linha mostra os resultados da instância

identificada na primeira coluna. São mostrados os valores da solução ótima (Exata) e do tempo execução do algoritmo *BT* em segundos (T). As colunas relativas aos algoritmos heurísticos mostram o valor da menor solução encontrada (Mín), a média das melhores soluções (Méd), o desvio padrão (σ), o afastamento percentual da melhor solução para a solução ótima (Δ%) e o tempo médio de execução em segundos (Tm).

Os resultados exibidos na Tabela 2 mostram que, de modo geral, o algoritmo *SABL* apresentou um desempenho superior. Ele encontrou a solução ótima de 49 das 50 instâncias, enquanto o algoritmo *SAop* conseguiu alcançar a solução ótima em 41 instâncias. O tempo médio de execução do *SAop* foi 0,15s e 0,64s para o *SABL*. O *SABL* também apresentou os menores valores médios e a menor variância dos resultados.

5. Conclusão

Este trabalho apresentou um novo problema de roteamento, chamado Problema do Caixeiro Viajante com Passageiros e Lotação (PCV-PL). Embora tenha como base o PCV, o PCV-PL possui características que sugerem para o modelo maior complexidade que o PCV com um maior número de decisões livres. A dicotomia entre priorizar a rota menos custosa ou buscar a máxima coleta de passageiros confere ao problema uma interessante característica de *trade off*. Observe-se que a coleta de passageiros é associada não só à sequência de localidades da rota quanto a restrições de tarifa dos próprios passageiros. Por outro lado, a capacidade do carro interfere na coleta e, consequentemente, também na rota ótima.

Como atividades futuras será realizada uma modelagem matemática, solução da mesma para diversas instâncias e desenvolvimento de novos algoritmos heurísticos.

Agradecimentos

Este trabalho foi financiado parcialmente pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), projetos 301845/2013-1 e 308062/2014-0.

Referências

Amey, A., Attanucci, J. & Mishalani. R. (2011). Real-Time Ridesharing – The Opportunities and Challenges of Utilizing Mobile Phone Technology to Improve Rideshare Services. *TRB Annual Meeting*, 1-17.

Balas, E., (2004). The prize collecting traveling salesman problem and its applications, in: G. Gutin e A. Punnen, editors, *The Traveling Salesman Problem and Its Variations, Combinatorial Optimization,* 12, Springer US, 663-695.

Calheiros, Z. S. A. (2015). *Algoritmos Evolucionários na Solução do Caixeiro Viajante com Caronas*, Monografia de Graduação, Universidade Federal do Rio Grande do Norte.

Dailey, D. J., Lose, D. & Meyers, D. (1999). Seattle smart traveler: dynamic ridematching on the world wide web. *Transportation Research Part C* 7(1):17-32.

Erdoğan, G., Cordeau, J-F. & Laporte, G. (2010). The Attractive Traveling Salesman Problem, *European Journal of Operational Research* 203:59-69

Feillet, D., Dejax, P. & Gendreau, M. (2005). Travelling salesman problems with profits, Transportation Science, 39:188-205.

Furuhata, M., Dessouky, M., Ordonez, F., Brunet, M., Wang, X. & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B* 57:28-46.

Garey, M. R. e Jonhson, D. S. (1979). Computers and intractability: a guide to the theory of NP-completeness*, W.H. Freeman and Co*.

Gendreau, M., Laporte, G. & Semet, F. (1998). A tabu search heuristic for the undirected selective travelling salesman problem, *European Journal of Operational Research* 106:539-545.

Gribkovskaiaa, I., Laporte, G. & Shyshou A. (2008). The single vehicle routing problem with deliveries and selective pickups*, Computers & Operations Research* 35(9):2908-2924.

Lin, S., Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem*, Operations Research,* 21.

López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T. & Birattari, M. (2011). The irace package, Iterated Race for Automatic Algorithm Configuration, Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics* 21(6):1087-1092.

Parragh, S. N., Doerner, K. F. & Hartl, R. F. (2008a). A survey on pickup and delivery problems - Part I: Transportation between customers and depot, *Journal für Betriebswirtschaft* 58(1): 21-51.

Parragh, S. N., Doerner, K. F. & Hartl, R. F. (2008b). A survey on pickup and delivery problems Part II: Transportation between pickup and delivery locations, *Journal für Betriebswirtschaft* 58(2): 81-117.

Vansteenwegen, P., Souffriau, W. & Van Oudheusden, D. (2011). The orienteering problem: A survey, *European Journal of Operational Research* 209:1-10.

Wang, X., Dessouky, M. & Ordóñez, F. (2013). A Pickup and Delivery Problem for Ridesharing Considering Congestion. *Transportation Letters: The International Journal of Transportation Research*.