Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 27, Num. 1 (2020) 62-82

RESEARCH ARTICLE

# Traveling Salesman Problem with Optional Bonus Collection, Pickup Time and Passengers

## Problema do Caixeiro Viajante com Coleta Opcional de Bônus, Tempo de Coleta e Passageiros

José Gomes Lopes Filho[1]*, Marco Cesar Goldbarg[1], Elizabeth Ferreira Gouvea Goldbarg[1], Vinícius Araújo Petch[1]

**Abstract:** This study introduces a variant of the Traveling Salesman Problem, named Traveling Salesman Problem with Optional Bonus Collection, Pickup Time and Passengers (PCVP-BoTc). It is a variant that incorporates elements of the Prize Collecting Traveling Salesman Problem and Ridesharing into the PCV. The objective is to optimize the revenue of the driver, which selectively defines which delivery or collection tasks to perform along the route. The economic effect of the collection is modeled by a bonus. The model can be applied to the solution of hybrid routing systems with route tasks and solidary transport. The driver, while performing the selected tasks, can give rides to persons who share route costs with him. Passengers are protected by restrictions concerning the maximum value they agree to pay for a ride and maximum travel duration. The activity of collecting the bonus in each locality demands a specific amount of time, affects the route duration, and is interconnected with the embarkment of passengers. Two mathematical formulations are presented for the problem and validated by a computational experiment using a solver. We propose four heuristic algorithms; three of them are hybrid metaheuristics. We tested the mathematical formulation implementations for 24 instances and the heuristic algorithms for 48.
**Keywords:** Travelling salesman Problem — Travelling salesman Problem with profits — Ridesharing

**Resumo:** O artigo introduz uma variante do Problema do Caixeiro Viajante, denominada de Problema do Caixeiro Viajante com Coleta Opcional de Bônus, Tempo de Coleta e Passageiros (PCVP-BoTc). É uma variante que incorpora elementos do Problema do Caixeiro Viajante com Coleta de Bônus e do *Ridesharing* ao PCV. O objetivo é otimizar as receitas do motorista que, seletivamente, define quais tarefas de entrega ou coleta serão executadas em sua rota. O efeito econômico da coleta é modelado através de um bônus. O modelo aplica-se na solução de sistemas híbridos de roteamento com tarefas de rota e transporte solidário. O motorista pode compartilhar assentos no carro de forma a ratear os custos de rota enquanto realiza suas tarefas selecionadas. Os passageiros são protegidos por restrições de valor máximo de rateio e tempo máximo de duração da viagem. A coleta de bônus demanda um tempo específico em cada localidade afetando a duração da rota e interligando-se com o embarque de passageiros. Duas formulações matemáticas são apresentadas para o problema e validadas através de um experimento computacional empregando um solver matemático. Quatro algoritmos heurísticos são propostos, sendo três algoritmos meta-heurísticos híbridos. As implementações das formulações foram testadas em 24 instâncias e os algoritmos sobre 48 instâncias.
**Palavras-Chave:** Caixeiro Viajante — Caixeiro Viajante com Coleta de Bônus — Passageiros

[1] *Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, Rio Grande do Norte, Brasil*
***Corresponding author**: zefilho@ppgsc.ufrn.br
**DOI:** http://dx.doi.org/10.22456/2175-2745.93733 • **Received:** 14/06/2019 • **Accepted:** 25/11/2019


## 1. Introdução

O Problema do Caixeiro Viajante (PCV) é um dos problemas mais investigados na Otimização Combinatória. Nele um caixeiro deseja determinar em um grafo ponderado $G = (V,A)$, no qual $V = \{1,\ldots,n\}$ representa o conjunto de vértices e $A = \{1,\ldots,m\}$ o conjunto de arestas, um ciclo Hamiltoniano de custo mínimo. O PCV é NP-Difícil [24] e possui inúmeras aplicações práticas em diversas áreas, como sistemas de manufatura, transportes, escalonamento de tarefas, roteiros de veículos, entre outros [21]. O Problema do Caixeiro Viajante

com Coleta de Bônus (PCV-CB) [6] e o Problema do Caixeiro Viajante com Passageiro (PCV-P) [8], são duas, dentre algumas variantes do PCV. No PCV-CB, um bônus é associado a cada vértice de $G$ e o objetivo do problema é encontrar o ciclo mais rentável considerando o balanço entre os custos da rota e os bônus coletados. No PCV-CB nem todas as localidades representadas no grafo $G$ são obrigatoriamente visitadas. O caixeiro só visita as localidades que selecionar para coletar o bônus, sendo penalizado pelas localidades que deixa de visitar. De acordo com Jozefowieze et al. [31], três problemas do PCV-CB podem ser definidos, dependendo da forma como os objetivos são colocados: i) *Profitable Tour Problem* (PTP), o objetivo desse problema é encontrar um circuito que maximize os bônus recolhidos subtraídos dos custos da viagem; ii) *Orienteering Problem* (OP), nesse problema o custo da rota é uma restrição e o objetivo do problema é encontrar um circuito que maximiza o total de bônus recolhido não excedendo o valor máximo pré-determinado do custo da rota; e por fim, iii) Problema do Caixeiro Viajante com Coleta de Prêmios (PCV-CP), nesse problema a coleta de bônus é uma restrição e o objetivo é encontrar a rota de menor custo, desde que o total de bônus recolhido não seja menor que o valor pré-determinado. No caso em que as penalidades do PCV-CP são nulas, o problema torna-se conhecido como Problema do Caixeiro Viajante com Quota (PCV-Q).

O Problema do Caixeiro Viajante com Passageiro (PCV-P) mescla o PCV com problemas de *ridesharing*, isto é, inclui a possibilidade de compartilhar o carro do caixeiro com passageiros de oportunidade e, assim, ratear as despesas de rota entre os ocupantes do carro. O transporte solidário apoia-se na ideia que há um acordo dos passageiros embarcados com o motorista em dividir as despesas entre si dos trechos compartilhados da viagem. Várias pesquisas mostram a importância dos impactos positivos do transporte solidário. Compartilhar assentos com passageiros de oportunidade pode auxiliar na diminuição da poluição ambiental e na redução no fluxo de carros nas vias de transporte, com a consequente redução nos engarrafamentos das metrópoles e melhoria da qualidade de vida. Adicionalmente, são visíveis os ganhos no conforto do passageiro e de sua socialização, na diminuição de custos dos operadores dos serviços, na redução das áreas de estacionamento de veículos e até nos investimentos públicos nos sistemas de transporte [14, 30, 34, 44].

O PCV-P é dado por um grafo $G = (V,A,P)$, no qual $P$ representa o conjunto de passageiros. Cada passageiro $p \in P$ tem origem na localidade $O_p \in V$ e destino $D_p \in V$, sendo $O_p \neq D_p$ e $T_p$ representa a despesa máxima que o passageiro $p$ aceita pagar. O passageiro $p$ ao ser embarcado, passa a dividir os custos, em rateio uniforme com os demais passageiros do veículo e motorista, calculando-se o rateio independentemente para cada trecho em que o passageiro seguir embarcado no veículo. O objetivo do PCV-P é de minimização, considerando que os custos de rota podem ser abatidos pelo rateio de despesas. O PCV-P é sujeito a restrições de capacidade do veículo, de precedência entre a localidade de embarque e a localidade de desembarque na rota, e o limite máximo de despesa admissível para cada um dos passageiros. Calheiros [8] demonstra que o PCV-P é NP-Completo. Gu et al. [22] investigaram versões mais simples de problemas que permitem o compartilhamento de caronas e provaram que são NP-difíceis mesmo quando apenas a origem, o destino e a capacidade do veículo são considerados.

Problemas que incorporam características do *ridesharing* ao seu núcleo como o PCV-P são estritamente relacionados com o Problema de Roteamento com Coleta e Entrega (PR-CE), que é uma generalização do Problema de Roteamento de Veículos, na qual objetos ou pessoas demandam transporte entre origens e destinos [7]. De acordo com Parragh et al. [38, 39] o PR-CE pode ser dividido em duas classes. A primeira se refere às situações que lidam com o transporte de mercadorias do depósito para os clientes e de clientes para o depósito. Já a segunda classe, agrupa os problemas em que as mercadorias são transportadas entre locais de coleta e entrega. De todas as especificações feitas pelos autores, os problemas *Dial-a-ride* (DARP) são os que modelam o comportamento do *ridesharing*. O DARP consiste em programar rotas e escalas de veículos de custos mínimos para atender pedidos de transporte de usuários, os quais especificam requisições de embarque e desembarque (conhecidos como pontos de coleta e entrega, respectivamente). No DARP o serviço de transporte é compartilhado no sentido de que vários usuários, com solicitações diferentes, podem estar no mesmo veículo ao mesmo tempo [28]. As pesquisas sobre DARP podem ser divididas em dois domínios: estático ou dinâmico. No caso estático, o motorista é atribuído ao um conjunto de requisições conhecidas de antemão. No dinâmico, as requisições dos passageiros chegam *on-line* e um motorista é atribuído a uma requisição que chega sem o conhecimento de requisições futuras [33]. Psaraftis [41] desenvolveu um dos primeiros estudos sobre o DARP com solicitação imediata, solucionado pequenas instâncias com um algoritmo de programação dinâmica. Uma revisão dos problemas dessa classe e métodos de soluções estão disponíveis nas referências [7, 10].

O atual trabalho introduz uma variante do PCV que considera características do PCV-CB e a existência da possibilidade de carregamento de passageiros como no PCV-P, denominado Problema do Caixeiro Viajante com Coleta Opcional de Bônus, Tempo de Coleta e Passageiros (PCVP-BoTc). Na variante examinada o objetivo é selecionar um subconjunto de localidades do problema que permita, considerando o embarque de passageiros, encontrar uma rota visitando as localidades uma única vez junto com um esquema de coleta de bônus que minimize a diferença entre os custos rateados da rota e o esquema de coleta de bônus. O modelo é sujeito a restrições de capacidade do veículo, do limite máximo de despesa para o passageiro e de um tempo máximo de permanência do passageiro embarcado no veículo. Os passageiros estão disponíveis para serem embarcados nas localidades somente dentro de uma janela de tempo fixada

individualmente pelo passageiro. Como um problema de minimização, a solução vazia é uma solução viável para o problema. A função objetivo do PCVP-BoTc se assemelha à do PTP proposta por Feillet et al. [16] com a inclusão da possibilidade de ratear os custos da rota do PCV-P. Diferentemente do problema de coletas de bônus formulado por Balas [6], o PCVP-BoTc não penaliza a ausência de visita a uma localidade, não obriga a coleta de um valor mínimo de bônus, e contabiliza, no tempo da rota, o tempo de cada coleta de bônus efetuada. O problema permite que o caixeiro opte por coletar ou não o bônus ao passar pela localidade. O caixeiro só acumula os tempos dos bônus coletados. Um embarque ao visitar uma localidade somente é possível caso o tempo da rota esteja dentro da janela de tempo do passageiro. Por fim, caso o caixeiro decida coletar o bônus de uma localidade, os possíveis passageiros só serão embarcados depois que a coleta for realizada. O desembarque dos passageiros em uma localidade antecede a coleta do bônus. Os passageiros não esperam dentro do veículo o bônus ser coletado para serem desembarcados, posteriormente, na mesma localidade.

O PCVP-BoTc encontra uma aplicação real no segmento de entregas expressas ou de *courier*. *Courier Services* são empresas que transportam e entregam documentos, pacotes e remessas de produtos. Tradicionalmente realizam entregas rápidas de documentos. Esta modalidade de negócio fornece serviços para empresas e indivíduos que precisam de serviço rápido, garantias de entrega e acompanhamento que o correio comum não aceita [40, 20].

A globalização econômica influencia praticamente todos os setores da economia. Os serviços de *Courier* não são uma exceção. Mesmo em seus mercados locais, as empresas de serviços estão sendo confrontadas por concorrentes globais. Como resultado, o ator do mercado local pode enfrentar, em sua região, uma concorrência internacional. Simultaneamente, a pressão para otimizar a qualidade e os custos dos serviços cresce. Assim, fornecer serviços econômicos e eficientes é cada vez de maior importância no setor de serviços [35]. O PCVP-BoTc pode potencializar a eficiência dos sistemas em serviços de *courier* produzindo impactos sociais e ecológicos positivos, bem como contribuindo para a melhoria do transporte urbano.

O presente artigo propõe duas formulações matemáticas para o problema e as soluciona através de um solver em um conjunto de instâncias especialmente formuladas para dar suporte aos estudos de *benchmark* dos algoritmos propostos neste estudo. São propostos quatro algoritmos: um algoritmo heurístico duas fases denominado NVH baseado em Programação Linear (PL), duas meta-heurísticas GRASP [17], e uma meta-heurística Colônia de Formigas [13]. As meta-heurísticas foram hibridizadas com técnicas VND (Variable Neighbourhood. Descent) [26] e métodos de Programação Linear. Os resultados alcançados através dos algoritmos heurísticos são comparados com as soluções encontradas para as formulações matemáticas.

Além da introdução, o artigo está organizado em mais 7 seções conforme segue. Na seção 2 é abordada a literatura relacionada ao problema. Na seção 3 é feita a descrição formal do problema. Duas formulações matemáticas são apresentadas na seção 4. Na seção 5 são apresentados os algoritmos propostos. Na seção 6 o banco de instâncias é descrito. Os experimentos computacionais são reportados na seção 7. Finalmente, as conclusões são expostas na seção 8.

## 2. Trabalhos Relacionados

Nesta seção serão descritos alguns trabalhos da literatura que possui conexão com a presente pesquisa. Os trabalhos abordam temas desde os problemas logísticos urbanos, como o compartilhamento de carona ou *ridesharing* [1], e o compartilhamento de recursos de transporte entre pessoas e mercadorias [33]; até as variantes clássicas do PTP.

O trabalho de Farhan e Chen [15] expõe o impacto do *ridesharing* na eficiência operacional de uma frota de veículos elétricos autônomos compartilhados (SAEVs), incluindo a determinação do tamanho da frota, locais da estação de carregamento, capacidade de atendimento da demanda de viagem e considerando os tempos de espera do usuário. Os autores propuseram um modelo de simulação baseado em agentes e que avalia as operações de uma frota de SAEVs em uma área metropolitana.

Alonso-Mora et al. [2] apresentaram um modelo matemático de alta capacidade para compartilhamento de caronas em tempo real, que é dimensionável para grandes números de passageiros e viagens, gerando dinamicamente rotas ótimas com relação à demanda *on-line* e localização de veículos. Os pesquisadores quantificaram experimentalmente a troca entre o tamanho da frota, capacidade, tempo de espera, atraso de viagem e custos operacionais para veículos de baixa a média capacidade, como táxis e van. O estudo experimental considera a possibilidade de caronas em veículos com capacidade de até dez pessoas.

Ma et al. [37], por sua vez, propuseram um modelo de compartilhamento de táxis que aceita solicitações de viagem em tempo real feitas por passageiros de táxis e enviadas meio de *smartphones*. O aplicativo otimiza a programação dos táxis de forma a atender os passageiros por meio do compartilhamento de assentos, que é sujeito a restrições de tempo, capacidade e monetária. As restrições monetárias proporcionam incentivos tanto para os passageiros quanto para os motoristas de táxi. O objetivo é minimizar a distância de viagem associada a cada táxi, enquanto atende aos pedidos de viagem.

Li et al. [33] expuseram, em seu estudo, os benefícios e desvantagens de combinar pessoas e fluxos de encomendas usando táxis e definiram o Problema da Partilha de Viagem (SARP). Os autores apresentaram formulações de MILP e realizaram um estudo numérico de cenários estáticos e dinâmicos para o problema.

Já Archetti et al. [4] definiram o Problema do Roteamento de Veículo com Motoristas Ocasionais (PRV-MO). No PRV-MO, uma empresa não possui somente uma frota de veículos e

motoristas disponíveis para fazer entregas, mas também pode se aproveitar de "motoristas ocasionais" (MOs) que estão dispostos a fazer uma única entrega usando seu veículo particular em troca de alguma compensação. Assim, a empresa procura satisfazer a desmanda dos clientes com custos totais mínimos, ou seja, os custos totais das rotas dos seus próprios veículos mais as compensações paga aos MO. Os autores fornecem informações valiosas sobre o potencial de usar motoristas ocasionais para reduzir os custos de entrega, concentrando-se principalmente no número e na flexibilidade dos motoristas ocasionais e do esquema de compensação empregado.

Arslan et al. [5] expuseram uma versão do PRV-MO dinâmico, enquanto Dahle et al. [11] apresentaram em seu trabalho uma extensão para o PRV-MO, em modelagem de coleta e entrega com janela de tempo e limite de tempo. Neste problema podem atender mais de uma solicitação e possuem uma restrição de compensação mínima para poder atender uma dada solicitação.

Archetti et al. [3] definiram o *Capacitated Profitable Tour Problem* (CPTP), no qual um único veículo com capacidade está disponível e o objetivo é maximizar a diferença entre o lucro total coletado e o custo da distância total percorrida. Handoko et al. [25] apresentaram o *Multi-vehicle Profitable Tour Problem* (MPTP). Tradicionalmente, o PTP envolve um único veículo. Entretanto, nesse trabalho os autores, consideram o PTP com vários veículos. Por sua vez, Sun et al. [43] introduziram o *Time-Dependent Capacitated Profitable Tour Problem With Time Windows And Precedence Constraints*, no qual o problema diz respeito à determinação de um passeio e seu tempo de partida no depósito que maximiza o lucro coletado menos o custo total de viagem (medido pelo tempo total de viagem).

Por outro lado, Zhang et al. [45] introduziram o *Probabilistic Profitable Tour Problem* (PPTP), no qual o problema é definido em um grafo completo e cada vértice (cliente) tem uma probabilidade de exigir uma visita em um determinado dia. O objetivo é encontrar um subconjunto de vértices e identificar, *a priori*, através desses vértices, um passeio, maximizando a diferença entre os lucros e os custos de viagem esperados.

Por fim, Lahyani et al. [32] apresentaram uma variação para o PTP denominada de *Rich Profitable Tour Problem* (RPTP) que surge quando solicitações de clientes envolvem vários produtos e veículos, utilizando vários compartimentos. No RPTP, o pedido de um cliente é composto por diferentes produtos. Um lucro está associado a cada produto. A demanda do cliente pode ser satisfeita parcialmente, entregando apenas alguns dos produtos listados na solicitação. Cumpre lembrar, entretanto, que circuitos viáveis são limitados no tempo e capacidade. Isso implica dizer que um veículo não pode visitar todos os clientes. Além disso, uma característica importante da variante é que alguns produtos são incompatíveis e devem ser mantidos separados durante o transporte. O veículo tem diferentes compartimentos com diferentes capacidades. São conhecidas incompatibilidade entre produtos e compartimentos. Por último, uma janela de tempo e um horário de atendimento estão associados a cada cliente. Os clientes podem ser coletados fora da janela de tempo mediante uma penalização na função objetivo. O custo total de uma excursão é obtido a partir do lucro total menos o custo da viagem e o custo dos tempos de espera.

De acordo com o nosso conhecimento, não foi encontrada nenhuma publicação que lide com todas as características combinadas do problema do presente estudo. Basicamente é uma variante que não pode lucrar com o compartilhamento de assentos no veículo, somente reduzir custos. Na próxima seção, será realizada uma descrição geral do problema.

## 3. Descrição do Problema

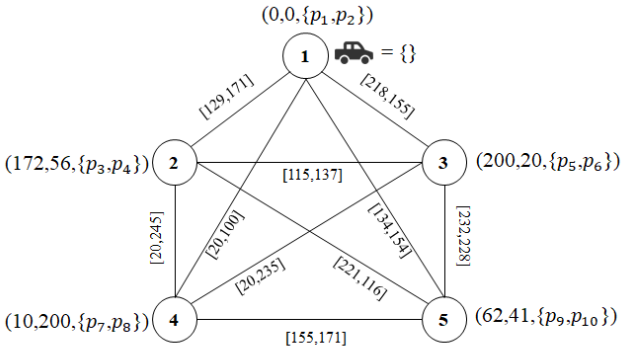
Nós consideramos um grafo $G = (V, A, P)$, sendo $V$ o conjunto de vértices, $A$ o conjunto das arestas, e $P$ o conjunto dos passageiros disponíveis em suas localidades. Um bônus não-negativo $b_i$ e um tempo gasto $s_i$ para o recolhimento desse bônus são associados a cada vértice $i \in V$. Custos $c_{ij}$ e tempos de percurso $t_{ij}$ são atribuídos a cada aresta. A cada $p \in P$ está associada uma origem $O_p \in V$, um destino $D_p \in V$, $O_p \neq D_p$, um valor $T_p$ representando a despesa máxima que $p$ está disposto a pagar, um tempo máximo de permanência no veículo $M_p$, e uma janela de tempo $J_p = [li_p, lf_p]$ representando um intervalo de tempo no qual o passageiro $p$ deseja que o compartilhamento da viagem seja iniciado. O caixeiro é o motorista do veículo com capacidade para $C$ passageiros. A capacidade não inclui o motorista. O objetivo do problema é encontrar um conjunto de localidades para serem visitadas uma única vez, um esquema de coleta de bônus, uma rota, e uma ordem de embarque de passageiros tais que minimizem a diferença entre os custos rateados da rota e o valor ganho com a coleta dos bônus. O PCV é um caso especial do PCVP-BoTc, no qual: 1. O menor bônus é duas vezes maior que a maior aresta de custo do grafo G. 2. todas as localidades do problema devem ser visitadas. 3. não existem passageiros para embarque. Portanto, para um caso geral, solucionar o PCVP-BoTc é pelo menos tão difícil quanto solucionar o PCV, clássico problema NP-Difícil [24].

A Figura 1 ilustra, em um grafo completo com cinco vértices, uma típica instância do PCVP-BoTc. No exemplo existe em cada vértice um rótulo $< a, b, c >$, onde $a$, $b$ e $c$ denotam, respectivamente, o bônus a ser coletado, o tempo de coleta e os passageiros disponíveis na localidade. Cada aresta também possui um rótulo, com a informação do custo e o tempo de travessia. A capacidade do veículo neste exemplo é $C = 3$. As requisições dos passageiros são representadas pelos rótulos $p_i =< O, D, T, M, li, lf >$, onde $O$ representa a origem do passageiro, $D$ o destino, $l$ a despesa máxima a ser paga, $M$ o tempo máximo de permanência no veículo e, $li$ e $lf$ o início e fim da janela de tempo para ser realizado seu embarque.

Uma solução ótima para o problema exemplificado na Figura 1 é exibida na Figura 2. Os vértices 2 e 3 e os pas-

sageiros $p_3$ e $p_6$ estão em destaque para representar que os bônus pertencentes aos vértices foram coletados e que os passageiros compartilharam trechos com o caixeiro. O vértice 4 pertence à solução, mas não está, o que em destaque representa que seu bônus não foi coletado.

A Figura 3 exemplifica os passos que o caixeiro percorreu na construção da solução. O caixeiro parte do vértice inicial e chega ao vértice 2, Figura 3a, coleta o bônus no vértice e permanece parado o tempo necessário para realizar o trabalho. Quando o caixeiro sai do vértice, o passageiro $p_3$ é embarcado para compartilhar o trajeto com o caixeiro até seu destino, e a capacidade do veículo é diminuída em uma unidade. O embarque do passageiro $p_3$, foi possível pelos fatos de o seu destino ainda não ter sido visitado, o tempo corrente da rota estar dentro da sua janela de tempo e existir um lugar disponível no veículo.




$p_1$ = (1, 2, 280, 571, 564, 847)
$p_2$ = (1, 5, 475,485,564,847)
$p_3$ = (2, 1, 412, 610, 0, 282)
$p_4$ = (2, 5, 489, 347, 282, 564)
$p_5$ = (3, 2, 287, 282, 0, 282)
$p_6$ = (3, 4, 643, 483, 0, 564)
$p_7$ = (4, 2, 478, 858, 282, 564)
$p_8$ = (4, 5, 371, 653, 564, 847)
$p_9$ = (5, 4, 539, 627, 564, 847)
$p_{10}$ = (5, 3, 490, 630, 0, 282)

**Figure 1.** Instância.

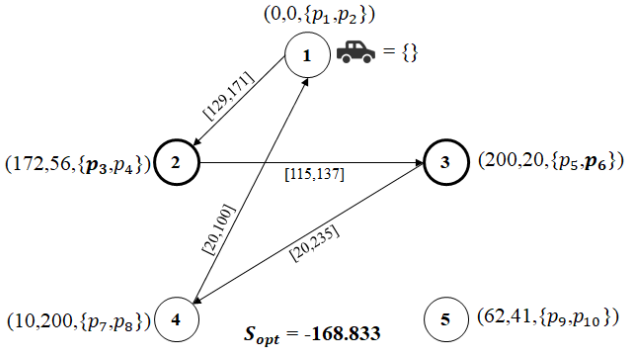



**Figure 2.** Uma solução ótima.

Em seguida, o caixeiro chega ao vértice 3, Figura 3b, e realiza a coleta do bônus, enquanto o passageiro $p_3$ fica à sua espera no carro uma vez que a localidade de seu desembarque não foi alcançada. O caixeiro parte do vértice 3 embarcando o único passageiro viável $p_6$ que passa a compartilhar o veículo com o caixeiro e $p_3$ até o seu destino. Com o embarque de $p_6$ a capacidade do veículo diminui outra unidade e passa a ser um. Chegando ao vértice 4, Figura 3c, o caixeiro realiza o desembarque do passageiro $p_6$ já que seu destino foi alcançado e aumenta uma unidade na capacidade do carro para, em seguida, examinar a possibilidade da coleta do bônus da localidade. O passageiro $p_6$ permaneceu embarcado até o seu destino uma vez que foram respeitadas todas as restrições a ele associadas. O caixeiro parte do vértice 4 sem realizar a coleta e continua com o passageiro $p_3$ embarcado. Por fim, o caixeiro chega ao vértice inicial, Figura 3d, e desembarca o passageiro $p_3$.

O custo total da solução do exemplo, corresponde ao custo $(c_{12} - b_2)$, somado com $(c_{23}/2 - b_3)$, sendo o custo de $c_{23}$ dividido por 2 pelo fato de o passageiro $p_3$ ter compartilhado esse trecho com o caixeiro, somado com $c_{34}/3$, sendo $c_{34}$ rateado por 3 pois $p_3$ e $p_6$ compartilharam esse trecho e finaliza com a soma de $c_{41}/2$, com o passageiro $p_3$ ainda embarcado. Assim, a solução final custa -168.833.

## 4. Formulações Matemáticas

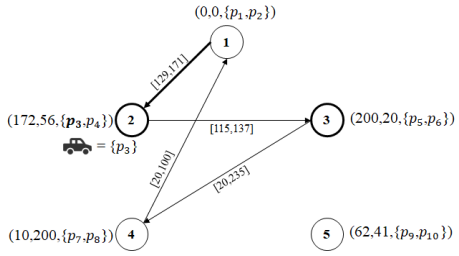
São apresentadas duas formulações não-lineares para o PCVP-BoTc baseadas em modelos de fluxo para o PCV. O primeiro modelo, apresentado na seção 4.1, é baseado na formulação de Gavish e Graves [19]. O segundo modelo, exposto na seção 4.2, estende a formulação proposta por Claus [9] que utiliza multifluxo. Ambos os modelos utilizam a função objetivo proposta por Feillet et al. [16] para PTP com uma simples modificação: o rateio nos trechos compartilhados com passageiros é considerado. Na seção 4.3, é apresentado um ajuste aos modelos permitindo que os dois sejam reescritos de forma quadrática.

### 4.1 Fluxo Simples (*FS*)

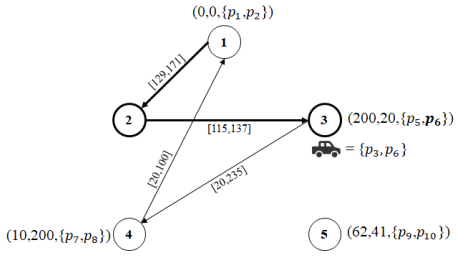
O modelo é baseado em uma formulação proposta por Gavish e Graves [19] para o PCV, com a adição de restrições relacionadas aos passageiros e os bônus. A função objetivo proposta em [16] para o PTP também é modificada. O problema é formulado em (1)-(24). Os parâmetros e as variáveis são definidas a seguir.

Parâmetros:

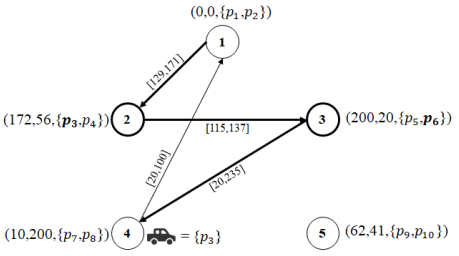
- $c_{ij}$: custo para atravessar a aresta $(i, j)$;
- $t_{ij}$: tempo para atravessar a aresta $(i, j)$;
- $b_i$: bônus no vértice $i$;
- $s_i$: tempo gasto para coleta do bônus no vértice $i$;
- $O_p$: origem do passageiro $p$;
- $D_p$: destino do passageiro $p$;



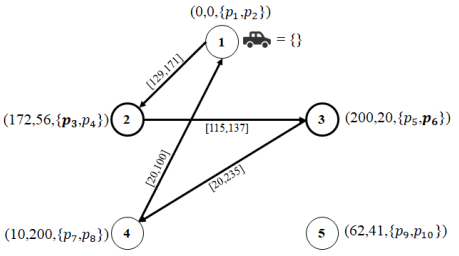

**(a)** Chegada do caixeiro no vértice 2.




**(b)** Chegada do caixeiro no vértice 3.




**(c)** Chegada do caixeiro no vértice 4.




**(d)** Chegada do caixeiro no vértice inicial.

**Figure 3.** Caminho trilhado pelo caixeiro na construção da solução da Figura 1.

- $T_p$: tarifa máxima que o passageiro $p$ admite pagar;
- $M_p$: duração máxima da viagem que o passageiro $p$ admite;
- $J_p = [li_p, lf_p]$: intervalo de tempo que o passageiro $p$ deseja ser embarcado;
- $C$: capacidade do veículo;
- $K$: constante relativamente grande.

Variáveis:

- $x_{ij}$: variável binária que indica se a aresta $(i, j)$ pertence à solução ($x_{ij} = 1$) ou não ($x_{ij} = 0$);
- $y_i$: variável binária que indica se o vértice $i$ está presente na solução ($y_i = 1$) ou não ($y_i = 0$);
- $z_i$: variável binária que indica se o bônus do vértice $i$ foi coletado ($z_i = 1$) ou não ($z_i = 0$);
- $v_{ij}^p$: variável binária que indica se o passageiro $p$ trafegou de $i$ para $j$ ($v_{ij}^p = 1$) ou não ($v_{ij}^p = 0$);
- $w_i^p$: variável binária que indica se o passageiro $p$ permaneceu esperando no vértice $i$ ($w_i^p = 1$) ou não ($w_i^p = 0$);
- $g_{ij}$: variável inteira que guarda a ordem em que a aresta $(i, j)$ foi visitada na rota;
- $f_{ij}$: variável inteira que guarda o tempo de chegada do caixeiro no vértice $j$ vindo de $i$.
- $r_i$: variável inteira que armazena o tempo de saída do caixeiro do vértice $i$.

$$\min \sum_{i \in V} \sum_{j \in V \setminus \{i\}} \frac{c_{ij} x_{ij}}{1 + \sum_{p \in P} v_{ij}^p} - \sum_{i \in V} b_i z_i \tag{1}$$

Sujeito a,

$$\sum_{j \in V \setminus \{i\}} x_{ij} = y_i \ , \forall i \in V \tag{2}$$

$$\sum_{j \in V \setminus \{i\}} x_{ji} = y_i \ , \forall i \in V \tag{3}$$

$$\sum_{j \in V \setminus \{i\}} g_{ij} - \sum_{j \in V \setminus \{i\}} g_{ji} = y_i \ , \forall i \in V \setminus \{1\} \tag{4}$$

$$g_{ij} \leq (|V| - 1) x_{ij} \ , \forall i, j \in V \tag{5}$$

$$z_i \leq y_i, \forall i \in V \tag{6}$$

$$\sum_{j \in V \setminus \{1\}} f_{1j} = \sum_{j \in V \setminus \{1\}} x_{1j} t_{1j} \tag{7}$$

$$\sum_{j \in V \setminus \{i\}} f_{ij} = \sum_{j \in V \setminus \{i\}} f_{ji} + \sum_{j \in V \setminus \{i\}} x_{ij} t_{ij} + z_i s_i \ , \forall i \in V \setminus \{1\} \tag{8}$$

$$f_{ij} \leq K x_{ij} \ , \forall i, j \in V \tag{9}$$

$$\sum_{j \in V \setminus \{O_p\}} v_{jO_p}^p + \sum_{j \in V \setminus \{D_p\}} v_{D_p j}^p = 0 \ , \forall p \in P \tag{10}$$

$$\sum_{j\in V\setminus\{i\}} v_{ji}^p - \sum_{j\in V\setminus\{i\}} v_{ij}^p = 0\ ,\forall p\in P, i\in V\setminus\{O_p,D_p\} \quad (11)$$

$$\sum_{j\in V\setminus\{1\}} v_{1j}^p + \sum_{j\in V\setminus\{1\}} v_{j1}^p = 0, \forall p\in P, O_p \neq 1\ e\ == D_p \neq 1 \quad (12)$$

$$\sum_{p\in P} v_{ij}^p \leq Cx_{ij}\ \ ,\forall i,j\in V \quad (13)$$

$$\sum_{i\in V}\sum_{j\in V\setminus\{i\}} \frac{c_{ij}v_{ij}^p}{1+\sum_{k\in P} v_{ij}^k} \leq T_p\ \ ,\forall p\in P \quad (14)$$

$$\sum_{i\in V}\sum_{j\in V\setminus\{i\}} t_{ij}v_{ij}^p + \sum_{i\in V} w_i^p s_i \leq M_p, \forall p\in P \quad (15)$$

$$\sum_{i\in V\setminus\{j\}} f_{ij} + z_j s_j = r_j, \forall j\in V\setminus\{1\} \quad (16)$$

$$li_p \sum_{j\in V\setminus\{O_p\}} v_{O_p j}^p \leq r_{O_p},\ \forall p\in P \quad (17)$$

$$r_{O_p} \leq lf_p + K\ (1\ -\ \sum_{j\in V\setminus\{O_p\}} v_{O_p j}^p), \forall p\in P \quad (18)$$

$$w_{O_p}^p + w_{D_p}^p = 0, \forall p\in P \quad (19)$$

$$2w_j^p \leq z_j + \sum_{i\in V} v_{ij}^p,\ \forall p\in P,\ j\in V\setminus\{O_p,D_p\} \quad (20)$$

$$w_j^p + 1 \geq z_j + \sum_{i\in V} v_{ij}^p,\ \forall p\in P,\ j\in V\setminus\{O_p,D_p\} \quad (21)$$

$$x_{ij}, y_i, z_i \in \{0,1\}\ \forall i,j\in V \quad (22)$$

$$v_{ij}^p, w_i^p \in \{0,1\}\ \forall i,j\in V, p\in P \quad (23)$$

$$r_i, f_{ij}, g_{ij} \in \mathbb{N}\ \forall i,j\in V \quad (24)$$

A função objetivo é expressa em (1). O primeiro e segundo termos da função objetivo, estão relacionados ao custo total do caixeiro compartilhado com os passageiros embarcados e os lucros recolhidos, respectivamente. As restrições (2) e (3) são conhecidas como restrições de atribuição, ou seja, garantem que todo vértice possuirá apenas uma aresta chegando e saindo dele, no circuito. As restrições (4) e (5) garantem que não existam subciclos e correspondem à formulação de fluxo simples para o PCV. A restrição (6) garante que o bônus da localidade só poderá ser recolhido caso esteja no circuito. As restrições (7) e (8) fazem com que a variável $f_{ij}$ acumule o tempo de percurso do caixeiro até o momento de chegada em determinada localidade. A restrição (9) garante que só haja tempo acumulado nos arcos em que o caixeiro trafegar. A restrição (10) inviabiliza o retorno de qualquer passageiro à sua origem, e de maneira análoga, inviabiliza o embarque de qualquer passageiro fora de sua localidade. A restrição (11) garante que se o passageiro for embarcado, seu desembarque é feito no seu destino. A restrição (12) inviabiliza que passageiros que possuem origem e destinos diferente da localidade inicial utilizem arestas relacionadas à localidade 1. A restrição (13) garante que as arestas utilizadas pelos passageiros não sejam diferentes das utilizadas pelo caixeiro e que o número de passageiros compartilhando o mesmo trajeto não seja maior do que a capacidade do veículo. As restrições (14) e (15) garantem que, para cada passageiro, o limite máximo do resultado do rateio e o tempo de permanência, caso sejam embarcados no veículo, não seja excedido. A restrição (16) relaciona o tempo de chegada e saída das localidades diferentes da inicial. As restrições (17) e (18) garantem que o passageiro só seja embarcado caso esteja no intervalo de sua janela de tempo. A valor de $K$ na restrição (18) foi definido como $K = \left\{\max_{i\in V}(s_i), \max_{i,j\in V}(t_{ij})\right\} * |V| * 2$ de modo que a desigualdade seja satisfeita para qualquer passageiro $p$ que não tenha sido embarcado.A restrição (19) garante que os passageiros não fiquem esperando o caixeiro coletar os lucros na sua origem e no seu destino. As restrições (20) e (21) garantem que, caso o caixeiro e os passageiros compartilhem trechos no circuito, os passageiros permaneçam embarcados durante o tempo da coleta do bônus da localidade, no caso de a localidade ser diferente da origem ou do destino do passageiro. Por fim, as restrições (22) e (24) asseguram o domínio das variáveis.

### 4.2 Formulação Baseada em Multifluxo (*MF*)

Nesta seção é apresentada uma segunda formulação para o problema derivada da formulação apresentada por Claus [9] que utiliza o conceito de multifluxo em redes para o PCV. Dado o grafo $G = (V,A,P)$, nós construímos um novo grafo $G^* = (V^*,A^*,P^*)$ no qual o vértice inicial é dividido em dois novos vértices, $q$ e $l$ representando a saída e chegada do caixeiro. Os bônus para ambos os vértices $q$ e $l$ são iguais ao do vértice inicial antes de ser dividido. Todos os arcos que chegam no vértice inicial são direcionados para $l$, enquanto todos os arcos de $G$ que partem do vértice inicial agora partem de $q$ em $G^*$. Um arco de $q$ para $l$ é adicionado em $G^*$ com o intuito de tratar a solução vazia, já que a mesma é possível no problema em questão. Arcos saindo do vértice $q$ para todos os outros vértices diferentes de $l$ são adicionados ao problema possuindo valores iguais a $\infty$. Arcos saindo do vértice $l$ para todos os outros vértices também são adicionados ao problema tendo $\infty$ com valores atribuídos. Todas as modificações são refletidas sobre o custo e tempo em cada aresta. Os passageiros em $G$ que têm como origem o vértice inicial passaram a ter $q$ como origem e os que tem como destino o vértice inicial passaram a ter $l$ como destino em $G^*$. Dessa forma, podemos interpretar o PCVP-BoTc como um problema de determinar um caminho de $q$ para $l$ no novo grafo que minimize o custo da rota menos o bônus coletado.

Assumindo que o vértice inicial é o 1, formalmente nós definimos $G^* = (V^*, A^*, P^*)$, como:

$$V^* = \{q, l\} \cup V \setminus \{1\}$$

$$\begin{aligned} A^* = A \setminus \{&(1,i),(i,1) : i \in V \setminus \{1\}\} \\ &\cup \{(q,l) := 0, (l,q) := \infty\} \\ &\cup \{(q,i) := (1,i) : i \in V \setminus \{1\}\} \\ &\cup \{(i,l) := (i,1) : i \in V \setminus \{1\}\} \\ &\cup \{(l,i) := \infty : i \in V \setminus \{1\}\} \\ &\cup \{(i,q) := \infty : i \in V \setminus \{1\}\} \end{aligned}$$

$$\begin{aligned} P^* = P \setminus &\{(O_p, D_p, T_p, M_p, J_p) : p \in P, O_p = 1 \; e \; D_p = 1\} \\ &\cup \{(O_p := q, D_p, T_p, M_p, J_p) : p \in P, O_p = 1 \; e \; D_p \neq 1\} \\ &\cup \{(O_p, D_p := l, T_p, M_p, J_p) : p \in P, O_p \neq 1 \; e \; D_p = 1\} \end{aligned}$$

O problema é formulado por (1), (10)-(11), (13)-(15), (17)-(21) e (25)-(36). Os parâmetros e as variáveis $x_{ij}$, $z_i$ , $v_{ij}^p$ , $w_i^p$, $r_i$ são os mesmos apresentados na seção anterior com a adição das seguintes variáveis.

Variáveis:

- $g_{ij}^k$:variável binária que controla se o fluxo $k$ trafega do vértice $i$ para o vértice $j$ ($g_{ij}^k = 1$) ou não ($g_{ij}^k = 0$);
- $y_i^k$: variável binária que indica se o fluxo $k$ coletou o bônus no vértice $i$ ($y_i^k = 1$) ou não ($y_i^k = 0$).

$$\sum_{j \in V^* \setminus \{q\}} x_{qj} + \sum_{j \in V^* \setminus \{l\}} x_{jl} = 2 \tag{25}$$

$$\sum_{j \in V^* \setminus \{q\}} x_{jq} + \sum_{j \in V^* \setminus \{l\}} x_{lj} = 0 \tag{26}$$

$$\sum_{j \in V^* \setminus \{k\}} x_{kj} - \sum_{j \in V^* \setminus \{k\}} x_{jk} = 0, \forall k \in V^* \setminus \{q, l\} \tag{27}$$

$$g_{ij}^k \leq x_{ij}, \forall k \in V^* \setminus \{q\}, \; i, j \in V^*, i \neq j \tag{28}$$

$$\sum_{j \in V^* \setminus \{k\}} g_{jk}^k = \sum_{j \in V^* \setminus \{k\}} x_{jk}, \forall k \in V^* \setminus \{q\} \tag{29}$$

$$\sum_{j \in V^* \setminus \{q\}} g_{qj}^k = \sum_{j \in V^* \setminus \{k\}} x_{jk}, \forall k \in V^* \setminus \{q\} \tag{30}$$

$$\sum_{j \in V^* \setminus \{i\}} g_{ij}^k - \sum_{j \in V^* \setminus \{i\}} g_{ji}^k = 0, \forall k, i \in V^* \setminus \{q\}, i \neq k \tag{31}$$

$$z_k \leq \sum_{j \in V^* \setminus \{k\}} x_{jk}, \forall k \in V^* \setminus \{q\} \tag{32}$$

$$2y_i^k \leq z_i + \sum_{j \in V^* \setminus \{i\}} g_{ji}^k, \forall k, i \in V^* \setminus \{q\} \tag{33}$$

$$y_i^k + 1 \geq z_i + \sum_{j \in V^* \setminus \{i\}} g_{ji}^k, \forall k, i \in V^* \setminus \{q\} \tag{34}$$

$$r_k = \sum_{i \in V^*} \sum_{j \in V^* \setminus \{i\}} t_{ij} g_{ij}^k + \sum_{i \in V^* \setminus \{1\}} y_i^k s_i, \forall k \in V^* \setminus \{q\} \tag{35}$$

$$y_i^k, g_{ij}^k \in \{0,1\} \; \forall i, j \in V^*, k \in V^* \setminus \{q\} \tag{36}$$

A função objetivo (1) e as restrições de (10)-(11), (13)-(15), (17)-(21) são as mesmas apresentadas na seção anterior com a simples mudança dos conjuntos $V$ por $V^*$ e $P$ por $P^*$. As restrições (25) e (26) garantem que o caminho inicie no vértice $q$ e termine no vértice $l$. A restrição (27) assegura a conservação do caminho do caixeiro. As restrições de (28) a (32) certificam que não haja subciclos e correspondem à formulação de multifluxo para o PCV proposto por Claus em [9]. A restrição (32) desempenha a mesma função que a restrição (6) do modelo anterior. As restrições (33) e (34) garantem se o fluxo $k$ coleta ou não o bônus no vértice $i$. A restrição (35) exerce a mesma função da restrição (16). Finalmente, a restrição (36) define o domínio das novas variáveis.

### 4.3 Formulação Quadrática (*FSQ* e *MFQ*)

O primeiro termo da função objetivo e a restrição (14) são não-lineares para ambos os modelos. Com objetivo de ajustar a formulação para eliminar a divisão em ambos os casos, o método proposto por [29] foi utilizado. A estratégia utilizada pelo método é transformar a divisão em uma multiplicação por um número real. Assim, para eliminar a operação de divisão, uma nova variável $\alpha_{ij}$ representando o valor inverso do denominador da divisão é introduzida permitindo que as formulações sejam reescritas de forma quadráticas. Adicionando as restrições (37) e (38) ao modelo *FS* a função objetivo (1) e a restrição (14) são reescritas em (39) e (40). O modelo *MF* também recebe as mesmas modificações levando em consideração os conjuntos $V^*$ e $P^*$.

$$\alpha_{ij} \times (1 + \sum_{p \in P} v_{ij}^p) = 1, \forall i, j \in V \tag{37}$$

$$0 \leq \alpha_{ij} \leq 1, \forall i, j \in V \tag{38}$$

$$\min \sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} x_{ij} \alpha_{ij} - \sum_{i \in V} b_i y_i \tag{39}$$

$$\sum_{i \in V} \sum_{j \in V \setminus \{i\}} c_{ij} v_{ij}^p \alpha_{ij} \leq T_p \;\; , \forall p \in P \tag{40}$$

Então, definimos a formulação quadrática de *FS* como *FSQ* consistindo da função objetivo (39) sujeito as restrições (2-13), (15-24), (37-38) e (40). Finalmente, *MFQ* representa a forma quadrática para o modelo *MF* consistindo da função objetivo (39) sujeito as restrições (10-11), (13), (15-21) e (25-36), (37-38) e (40).

## 5. Algoritmos Propostos

Esta seção descreve quatro algoritmos propostos neste estudo. O primeiro algoritmo é uma heurística que decompõe o problema em duas etapas. Cada uma das etapas é solucionada através de uma ferramenta de otimização. O segundo e terceiro são algoritmos baseados na meta-heurística GRASP. A função dos algoritmos descritos é estabelecer soluções para as instâncias examinadas de forma a testar a eficiência do quarto algoritmo proposto que é uma meta-heurística Colônia de Formigas. Todas as meta-heurísticas são hibridizadas, conforme descrito adiante.

Cada solução, $S$, é representada por dois vetores como ilustrado na Figura 4. O vetor $R$ representa a sequências de visitas do caixeiro com a opção de coleta do bônus. Cada posição do vetor $R$ é formada por uma dupla $(x,y)$, onde $x$ representa a localidade visitada e $y$ a opção de coleta. O valor de $y$ é 1 caso o bônus seja coletado e 0 caso contrário. O segundo vetor, $P$, representa os passageiros que foram embarcados. O ciclo inicia e termina sempre na localidade 1, mas ela não é representada como destino final na solução.

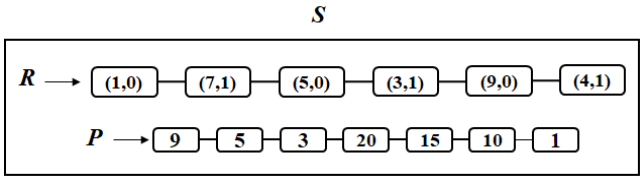



**Figure 4.** Representação da solução para o PCVP-BoTc.

### 5.1 NVH

O algoritmo 1 apresenta a heurística denominada NVH, que é dividida em duas fases: a primeira, encontra uma rota para um conjunto de localidades pré-selecionadas. A segunda encontra um carregamento para a rota. A duas fases são solucionadas através de uma ferramenta de otimização. O algoritmo possui três parâmetros de entrada: *nomeInst*, o nome da instância; *tempoRota*, o tempo máximo em segundos, para o *solver* resolver o subproblema de rota; e *tempoPass*, o tempo máximo, em segundos, para o *solver* solucionar o carregamento de passageiros da rota.

O algoritmo define uma solução vazia na linha 3. O passo 4 é a primeira fase da heurística, resolverRota(), implementando as restrições de (2)-(6) e a função objetivo (1) sem o rateio dos custos da formulação apresentada na seção 4. A função recebe, $G$, o grafo do problema e *tempoRota*. O retorno da função é um circuito com o esquema de coleta sem os passageiros que foi encontrado em no máximo *tempoRota* segundos pelo otimizador.

O passo 5 implementa a segunda fase, função embarcarPass(), que resolve a atribuição de passageiros para a rota preestabelecida, fixando valores para as variáveis $x_{ij}$, $y_i$ e $z_i$ do modelo proposto *FSQ*. Desse modo, a formulação é reduzida às restrições (7)-(13),(15)-(21), (37)-(38) e (40) com a função objetivo (39). O procedimento recebe como parâmetro, o grafo $G$, a rota preestabelecida $S.R$, e o *tempoPass*. A função retorna a melhor configuração de passageiros encontrada pelo otimizador em no máximo *tempoPass* segundos para a rota preestabelecida $S.R$.

Por fim, o algoritmo retorna uma solução completa.

**Algoritmo 1:** NVH

**Entrada:** *nomeInst*, *tempoRota*, *tempoPass*
**Saída:** $S = (R,P)$

```
1 início
2     G ← leituraInst(nomeInst) ;
3     S ← ∅;
4     S.R ← resolverRota(G, tempoRota) ;
5     S.P ← embarcarPass(G, S.R, tempoPass);
6     retorna S;
7 fim
```

### 5.2 GVND1

Esta seção apresenta o algoritmo GRASP hibridizado com uma busca VND (*Variable Neighborhood Descent*), com a etapa de carregamento de passageiros realizada por um solver. O pseudocódigo do GVND1 é apresentado no algoritmo 2. São quatro parâmetros de entrada: *nomeInstancia*, o nome da instância, *maxFc*, o número máximo de chamadas à função objetivo realizáveis pelo algoritmo, *tamLista*, o tamanho da lista restrita de candidatos e, por fim, *maxConv*, o número máximo de iterações sem melhoria.

O algoritmo carrega a instância através de leituraInstancia() e define as variáveis $S$, *numCh* e *conv*. $S$ representa uma solução; $f(S)$ é o valor da solução; *numCh* conta o número de chamadas da função objetivo e *conv* é um contador para o número de iterações sem melhoria. O laço principal tem dois critérios de parada, quais sejam, o número de chamadas à função objetivo e o número de iterações sem melhoria.

A construção da solução $S^*$ é realizada pelo método constrSolucao(), cujo pseudocódigo é expresso pelo algoritmo 3. A solução encontrada é submetida à fase de VND, função VND(), descrita no algoritmo 4. Finalizando, caso haja melhoria a solução $S$ é atualizada.

A constrSolucao($G$, *tamLista*) é inicializada com o grafo da instância e com o tamanho da lista restrita de candidatos. O procedimento constrói iterativamente soluções cujo comprimento varia de 1 a $|V|$, preservando a melhor solução. O procedimento é iniciado na localidade 1. A solução inicial tem tamanho 1 (um) e valor objetivo 0 (zero) representando a solução vazia $S$. Em seguida, é formada uma lista de candidatos *LCD* utilizando o método montarLCD(*cidadeatual*) a partir da localidade corrente. A lista *LCD* contém todas as possibilidades de destino $j$ da localidade corrente $i$ que podem ser incluídas na solução. Cada vértice destino $j$ contribuirá em duas possibilidades para *LCD* de acordo com as equações (41) e (42). A primeira opção representada na equação (41) mensura a distância do vértice $i$ para $j$ como a soma do custo de

**Algoritmo 2:** GVND1

**Entrada:** *nomeInst*, *tamLista*, *maxFc*, *maxConv*
**Saída:** $S = (R,P)$

**início**
  $G \leftarrow$ leituraInst(*nomeInst*) ;
  $S \leftarrow \emptyset$; $f(S) \leftarrow \infty$;
  $numCh \leftarrow 0$;
  $conv \leftarrow 0$;
  **enquanto** $(numCh \leq maxFc)$ **e** $(conv \leq maxConv)$ **faça**
    $S^* \leftarrow$ constrSolucao($G$, *tamLista*);
    $S^* \leftarrow$ VND($S^*$);
    **se** $f(S^*) < f(S)$ **então**
      $S \leftarrow S^*$;
      $conv \leftarrow -1$;
    **fim**
    $conv \leftarrow conv + 1$;
  **fim**
  **retorna** $S$;
**fim**

percorrer as arestas $(i, j)$ e $(j, 1)$ menos o valor do bônus coletado no vértice $j$. A segunda opção representada na equação (42) mensura uma segunda distância da localidade $i$ para $j$.

**Algoritmo 3:** constrSolucao

**Entrada:** $G$, *tamLista*
**Saída:** $\pi = (R,P)$

**início**
  $\pi \leftarrow \emptyset$; $f(\pi) \leftarrow \infty$;$\pi^* \leftarrow \emptyset$; $f(\pi^*) \leftarrow \infty$;
  $cidadeAtual \leftarrow 1$;
  $\pi^*.R.inserir(< cidadeAtual, 0 >)$;$f(\pi^*) \leftarrow 0$;
  **enquanto** $|\pi^*.R| \leq |G.V|$ **faça**
    $LCD \leftarrow$ montarLCD($cidadeAtual$);
    $LRC \leftarrow$ montarLRC(*LCD*, *tamLista*);
    $cidadeAtual$, $opCol \leftarrow$ sorteioLRC(*LRC*);
    $\pi^*.R.inserir(< cidadeAtual, opCol >)$;
    $\pi^*.P \leftarrow$ embarcarPass($G$, $\pi^*.R$, 60);
    **se** $f(\pi^*) < f(\pi)$ **então**
      $\pi \leftarrow \pi^*$;
    **fim**
    $numCh \leftarrow numCh + 1$;
    testarCh($numCh$,$\pi$);
  **end**
  **retorna** $\pi$;
**fim**

Dado que a primeira opção de um destino $j$ foi sorteada para compor a solução, o caixeiro passará pela localidade $j$ e coletará o bônus. Caso a segunda seja sorteada o caixeiro passará pela localidade e não coleta o bônus.

Considerando que a lista *LCD* esteja completa, o próximo passo é montar a lista *LRC* através do método montarLRC(*LCD*, *tamLista*) que recebe como entrada a lista de candidatos e o tamanho da lista *LRC*. A lista *LRC* contém *%tamLista* das localidades com os menores valores de *LCD*. Em seguida, o destino $j$ é sorteado aleatoriamente na lista *LRC*, armazenado na variável *cidadeAtual*, e a opção de coleta é atribuída à *opCol*. Assim, a dupla (*cidadeAtual*, *opCol*) é inserida no final da solução corrente. A função embarcarPass($G$, $\pi^*.R$, 60) é acionada para viabilizar o carregamento da nova rota. Essa função é equivalente à que foi apresentada no algoritmo NVH. A nova solução é comparada com a melhor a solução já existente, preservando-se a melhor. A variável *numCh* é incrementada nas iterações do laço e o procedimento testarCh(*numCh*, $\pi^*$) verifica se o número de chamadas não excedeu o número máximo de chamadas da função objetivo *maxFC*. Caso o valor de *numCh* seja maior que *maxFC* o algoritmo compara $\pi^*$ com a melhor solução encontrada, atualizando a solução $\pi$, se for o caso, retornando o resultado. O procedimento de construção continua até o tamanho da rota alcançar o previsto. Ao final do procedimento é retornada a solução com melhor valor objetivo.

$$D^1_{ij} = c_{ij} + c_{j1} - b_j \tag{41}$$

$$D^2_{ij} = c_{ij} + c_{j1} \tag{42}$$

O passo seguinte realiza a busca local GRASP no formato de uma busca local VND, resumida no algoritmo 4. O VND recebe como entrada $\pi$, a solução corrente, que será perturbada. $\xi$ representa um conjunto de vizinhanças da busca VND. As estruturas de vizinhanças aplicadas, em ordem, foram:

- Pass1: seleciona um passageiro $p$ que não possui origem ou destino de maior bônus na rota corrente e insere a localidade origem/destino com a opção de coletar o bônus ativada na rota corrente. A inserção, caso o vértice seja a origem, é feita em todas as posições anteriores ao destino. Caso seja a localidade de destino, a inserção é realizada em todas as posições depois da origem;
- Pass2: seleciona um passageiro $p$ que não possui origem e destino na rota corrente com maior valor de $b_{O_p} + b_{D_p} - c_{O_pD_p}$e insere a localidade origem e destino em posições consecutivas, ambos com a opção de coletar o bônus ativada, testando todas as posições da solução.
- Inserir: insere o vértice de maior bônus que não está na rota corrente em todas as posições com a opção de coleta ativada;
- Troca: seleciona uma posição aleatória da solução e realiza a troca com todas as outras;
- Inverte: inverte o caminho de todas as posições até o final da rota;

- Apagar: apaga um vértice da rota. Esta vizinhança é executada para todas as posições da rota corrente menos a que contém o vértice inicial;
- Exato: seleciona três posições consecutivas na rota e testa todas as combinações possíveis sobre esses três vértices e repete para todas as outras posições que pertencem à rota corrente;
- $K$-opt: troca $k$ arestas sobre o subgrafo induzido pela rota corrente através do algoritmo de Helsgaun [27] denominado LKH-2.

**Algoritmo 4:** VND

```
   Entrada: π, ξ
   Saída: π = (R,P)
1  início
2      melhoria ← 1;
3      enquanto melhoria faça
4          melhoria ← 0;
5          k ← 1;
6          enquanto k ≤ |ξ| faça
7              π* ← buscaLocal(π,k);
8              se f(S*) < f(S) então
9                  π ← π*;
10                 melhoria ← 1;
11             senão
12                 k ← k+1;
13             fim
14         fim
15     fim
16     retorna π;
17 fim
```

A sequência da aplicação das vizinhanças do VND foi organizada de forma a priorizar a busca em vizinhanças mais eficientes. A partir da solução inicial $\pi$, o método explora as estruturas de vizinhança segundo sua prioridade pré-estabelecida. O algoritmo progride na lista quando a vizinhança examinada falha em encontrar uma melhoria para a solução corrente. Alcançada a última vizinhança da lista e não ocorrendo melhoria, o algoritmo finaliza a etapa VND.

As vizinhanças aplicam perturbações na solução corrente. Logo, para manter a viabilidade dos passageiros, a função embarcarPass($G$, $\pi^*.R$, 60) é executada sobre cada perturbação com intuito de reparar a atribuição dos passageiros na nova rota. A cada execução de embarcarPass() a variável *numCh* é incrementada em uma unidade e o procedimento testarCh() é executado. Por fim, o algoritmo termina quando o número de chamadas à função objetivo atingir *maxFc* ou quando a melhor solução não é melhorada em *maxConv* iterações. A melhor solução é retornada quando um dos critérios de parada é satisfeito.

### 5.3 GVND2

O GVND2 é uma versão do GVND1 com um novo método de construção da solução. O algoritmo 5 apresenta o pseudocódigo do novo método para a fase construtiva. Ele recebe como parâmetros de entrada $G$; *tamLista*, o tamanho da lista restrita de candidatos; e *listaOrdP*, uma lista contendo os passageiros ordenados de acordo com o início da janela de tempo de cada um. O algoritmo inicia definindo uma solução $\pi$ e insere na sua rota o vértice inicial com a opção de coleta desativada (0), atribuindo seu valor objetivo como 0. Uma segunda solução $\pi^*$ igual a $\pi$ é criada para servir de auxiliar no decorrer do procedimento. Em seguida, o procedimento entra no laço principal que é executado enquanto houver passageiros em *listaOrdP*. Logo após, é formada uma lista de candidatos usando o método montarLCD($\pi^*$, *listaOrdP*) que recebe a solução $\pi^*$ e a lista de passageiros *listaOrdP* como parâmetros para montar a lista com todos os passageiros cujas localidades de origem e destino ainda não estejam na solução $\pi^*$. A lista é armazenada em *LCD*. A localidade de montarLCD() é tratada como se não estivesse na solução $\pi^*$. Considerando que a lista *LCD* esteja completa o próximo passo é montar a lista *LRC* através do método montarLRC(*LCD*, *tamLista*) que recebe *LCD* e o tamanho que a *LRC* terá. A *LRC* contém todos *%tamLista* passageiros com os menores valores de início da janela de tempo em *LCD*. Posteriormente, um passageiro, *pass*, é sorteado aleatoriamente em *LRC* pelo método sorteioLRC(*LRC*).

Com o passageiro sorteado, a linha 9 realiza um teste utilizando o método $testarJanela(\pi^*, pass)$ para verificar se o tempo total da solução $\pi^*$ com a inserção no final da rota da localidade origem de *pass* e a opção de coleta ativada permanece dentro da janela de tempo do passageiro *pass*. Um caso especial é considerado no teste quando a localidade origem de *pass* é a localidade 1, o retorno só será positivo se o início da janela de tempo de *pass* for igual a 0. Sendo positivo o resultado do método $testarJanela()$, o passageiro *pass* está apto a compartilhar algum trecho com o caixeiro. O trecho é criado pelo método expandirPass() levando em considerando três casos possíveis:

1. o primeiro caso acontece quando o passageiro *pass* possui como vértice origem $O_{pass} = 1$ e vértice destino $D_{pass} \neq 1$. Esse evento só ocorre quando a rota da solução corrente possui somente o vértice inicial. Como o vértice origem de *pass* é o vértice inicial e já pertence à solução $\pi^*$, então não precisará ser inserido na rota. Em seguida, são realizadas inserções sucessivas de vértices de forma gulosa entre $O_{pass}$ e $D_{pass}$, tentando aumentar o caminho entre a origem e o destino de *pass*. O procedimento inicia e busca pelo vértice pertencente a $G$ que não está na rota da solução corrente $\pi^*$, que seja diferente de $D_{pass}$, que possua o maior bônus cuja inserção antes do vértice destino de *pass* respeite as restrições de tarifa e cujo tempo máximo de permanência seja respeitado. O teste das restrições de despesa e tempo máximo é feito na medida em que o

**Algoritmo 5:** constrSolucaP

```
   Entrada: G, tamLista, listaOrdP
   Saída: π = (R,P)
1  início
2      π ← ∅;
3      π.R.inserir(< 1,0 >); f(π) ← 0;
4      π* ← π;
5      enquanto |listaOrdP| > 0 faça
6          LCD ← montarLCD(π*, listaOrdP);
7          LRC ← montarLRC(LCD, tamLista);
8          pass ← sorteioLRC(LRC);
9          se testarJanela(π*, pass) então
10             expandirPass(G, π, π*, pass);
11             se D_pass = 1 então
12                 retorna π;
13             fim
14         senão
15             se expandirRota(G, π, π*, pass) então
16                 expandirPass(G, π, π*, pass);
17                 se D_pass = 1 então
18                     retorna π;
19                 fim
20             fim
21         fim
22         atualizarLista(π*, listaOrdP);
23     fim
24     retorna π;
25 fim
```

caminho entre a origem e destino de *pass* é expandido com o acúmulo das arestas $c_{ij}$ e $t_{ij}$. A localidade $j$ respeita as restrições se a sua inserção no caminho entre o destino de *pass* e o vértice anterior atender as restrições: $(c_{O_{pass}i} + \cdots + c_{jD_{pass}}) \leq T_{pass}$ para despesa máxima e $(t_{O_{pass}i} + \cdots + t_{jD_{pass}}) \leq M_{pass}$ para o tempo máximo de permanência. O procedimento só retorna a solução modificada quando não houver mais vértices que possam ser inseridos entre a origem e o destino de *pass*. O procedimento expandirPass() termina inserindo a localidade $D_{pass}$ no final da rota para fechar o caminho expandido entre a origem e o destino de *pass*. Todas as inserções desempenhadas pelo método são realizadas com a opção de coleta de bônus ativada. No decorrer da execução de expandirPass() o procedimento testa se a solução $\pi^*$ tem valor objetivo melhor que $\pi$ e atualiza $\pi$ se for o caso. Finalizando, o procedimento retorna as soluções modificadas;

2. o segundo caso acontece quando o passageiro *pass* possui localidades de origem e destino diferentes da localidade inicial. Dessa forma, a localidade origem é inserida na rota. Depois, o método tenta expandir o caminho entre a origem e o destino de *pass* inserindo vértices entre eles e finaliza com a cidade destino;

3. o terceiro caso acontece quando o passageiro *pass* possui vértice $O_{pass} \neq 1$ e o destino de *pass* $D_{pass} = 1$. Logo, somente o vértice origem será inserido na rota na solução corrente. Posteriormente, o procedimento tenta aumentar o caminho entre a origem e o destino de *pass* e finaliza retornando as soluções modificadas.

Analogamente aos procedimentos já descritos, as modificações realizadas em expandirPass() mantêm os passageiros viáveis. *numCh* é atualizada e o método testarCh() é executado. Finalizado a execução de expandirPass() na linha 10, o passo 11 testa se o destino de *pass* é o vértice inicial. Caso positivo a melhor solução encontrada até o momento é retornada e a etapa de construção da solução é encerrada.

Seguindo a execução de constrSolucaoP(), o procedimento continua sua execução na linha 14 caso o teste expresso na linha 9 retorne um valor negativo. Assim, a linha seguinte é executada e o procedimento *expandirRota*() é chamado sobre o teste. Esse procedimento é semelhante a expandirPass(). O objetivo de expandirRota() é inserir vértices, também de forma gulosa, na solução corrente na tentativa de enquadrar o tempo total da rota com a janela de tempo do passageiro *pass*, tornando-o apto ao embarque no veículo. A função gulosa para esse método possui a seguinte característica: dado que $i$ seja o último vértice da rota da solução corrente $\pi^*$, o vértice $j$ diferente da origem/destino de *pass* que possuir a maior proporção $b_j/c_{ij}$ será o candidato a ser inserido na rota de $\pi^*$.

À medida que as inserções são realizadas, o tempo total da rota é incrementado e testado com a janela de tempo de *pass* para verificar se houve sobreposição do tempo sobre a janela. As inserções deixam de ser realizadas no momento em que o tempo total da rota se enquadra na janela de tempo de *pass*, quando não há mais vértices disponíveis ou no momento em que o tempo total da rota ultrapassar o final da janela de tempo de *pass*. O procedimento retorna positivo se e somente se o tempo total da rota estiver dentro da janela de tempo de *pass*. Toda modificação na solução realizada em expandirRota() é refletida em $\pi^*$. No seu decorrer, esse método viabiliza os passageiros, atualiza *numCh*, executa testarCh() e compara se a solução $\pi^*$ tem valor objetivo melhor que $\pi$ e atualiza $\pi$ se necessário. Caso o teste na linha 15 seja positivo, o trecho da linha 16 a 19, semelhante a outros já apresentados anteriormente, é executado. Como o passageiro foi viabilizado, o vértice origem é inserido com a opção de coleta e o método expandirPass() é chamado com o intuito de aumentar o caminho entre $O_{pass}$ e $D_{pass}$. O procedimento se encerra se o destino de *pass* for o vértice inicial, caso contrário a execução continua.

Por fim, na linha 22 o método atualizarLista(*listaOrdP*, $\pi^*$) é executado retirando todos os passageiros que possuem suas localidades origem/destino presentes na rota da solução $\pi^*$ ou passageiros cujas janelas de tempo foram ultrapassadas pelo tempo total da rota de $\pi^*$. Em seguida, uma nova iteração do laço é executada. O procedimento de construção

da solução, constrSolucaoP(), encerra e retorna a melhor solução quando a lista de passageiros, *listaOrdP*, estiver vazia ou quando as linhas 12 e 18 forem executadas.

### 5.4 Otimização por Colônia de Formigas - ACO

A meta-heurística ACO foi proposta inicialmente por Dorigo e Di Caro [13]. Desde então, os algoritmos baseados em ACO têm sido aplicados em diversos problemas, tal como o PCV, roteamento de veículos, coloração de grafos e outros.

No algoritmo ACO, cada formiga representa uma solução completa que, no caso do PCVP-BoTc, é um circuito com sua coleta de bônus e os passageiros embarcados. Esta seção apresenta um algoritmo ACO híbrido para o problema. O pseudocódigo do ACO é apresentado no algoritmo 6. São fornecidos oito parâmetros de entrada: *nomeInstancia*, a instância a ser solucionada; *maxFc*, o número máximo de chamadas da função objetivo que o algoritmo pode realizar; *maxConv*, o número máximo de iterações sem melhoria que o algoritmo pode considerar; $\rho$, taxa de evaporação do feromônio; $\alpha$, controle da influência do feromônio; $\beta$, controle de influência da informação heurística da rota; $\gamma$, controle da influência da informação heurística dos passageiros; *numF*, o número de formigas e $\omega$, a influência que a solução global e local desempenharão na atualização dos feromônios.

Na inicialização, o algoritmo ACO representa o problema segundo o modelo $MF$. A inicialização dos feromônios é realizada na linha 3. Para cada arco $i-j$ é atribuída uma dupla $\tau_{ij} = (\tau^1_{ij}, \tau^2_{ij})$, onde $\tau^1_{ij}$ representa a quantidade de feromônio associado ao arco $i-j$ com o bônus no vértice $j$ sendo coletado e $\tau^2_{ij}$ a quantidade de feromônio também associado ao arco $i-j$ sem que bônus no vértice $j$ seja coletado. O índice $k$ de $\tau^k_{ij}$ faz referência à possibilidade de bônus ser coletado ou não no vértice $j$. O valor inicial dos feromônios é $1/|V|$. As variáveis $S$, *numCh*, e *conv* desempenham o mesmo papel apresentado no algoritmo GVND1. Em seguida, duas soluções $S'$ e $S''$ são construídas com o objetivo de alimentar a tabela de feromônios. $S'$ é gerada por constrSolucaoPG(G) que é uma versão gulosa do algoritmo construSolucaoP(). O procedimento ordena os passageiros e a construção da solução se dá escolhendo o passageiro com a janela de tempo com tempo mais precoce. $S''$ é concebida pelo algoritmo NVH descrito no algoritmo 1. O solver é executado por até 60 segundos na rota e outros 60 segundos no carregamento. O algoritmo VND é aplicado às soluções $S'$ e $S''$ como descrito no algoritmo 4 visando refinar as soluções. A partir das soluções alcançadas pelo VND, o passo 10 executa aumentarFero() com objetivo de duplicar a quantidade de feromônio $\tau^1_{ij}$ ou $\tau^2_{ij}$ nas arestas $i-j$ utilizando as soluções $S'$ e $S''$. $\tau^1_{ij}$ será atualizado se o arco $i-j$ pertencer a alguma das soluções e o bônus no vértice $j$ tenha sido coletado, caso o bônus em $j$ não tenha sido coletado $\tau^2_{ij}$ que será atualizado.

O laço principal do algoritmo inicia na linha 11 e possui os mesmos critérios de parada dos algoritmos GVND. O laço no passo 13 cria *numF* formigas. A construção das formigas é realizada pelo método constrForm($G$, $\tau$) que consulta a

**Algoritmo 6:** ACO

**Entrada:** *nomeInst*, *tamLista*, *maxFc*, *maxConv*, $\rho$, $\alpha$, $\beta$, $\gamma$, *numF*, $\omega$
**Saída:** $S = (R, P)$

```
1  início
2      G ← leituraInst(nomeInst);
3      τ ← iniciarFero(G);
4      S ← ∅; f(S) ← ∞;
5      numCh ← 0; conv ← 0;
6      S' ← constrSolucaoPG(G);
7      S' ← VND(S');
8      S'' ← NVH( "60s", "60s");
9      S'' ← VND(S'');
10     aumentarFero(τ, S', S'');
11     enquanto
         (numCh ≤ maxFc) e (conv ≤ maxConv) faça
12         itF ← 1; S* ← ∅; f(S*) ← ∞;
13         enquanto itF ≤ numF faça
14             S' ← constrForm(G,τ);
15             se f(S') < f(S*) então
16                 S* ← S';
17             fim
18             itF ← itF + 1;
19         fim
20         S* ← VND(S*);
21         se f(S*) < f(S) então
22             S ← S*;
23             conv ← −1;
24         fim
25         conv ← conv + 1;
26         atualizarFero(τ, ρ, S, S*);
27     fim
28     retorna S;
29 fim
```

tabela de feromônios. Nesta etapa de construção, todas as formigas iniciam na localidade inicial, $q$, e constroem rotas de modo incremental. A construção da rota se encerra quando o vértice $l$ é sorteado. Em cada etapa da construção, cada formiga localizada no vértice $i$ usa duas regras de transição para selecionar o próximo vértice $j$ com objetivo de equilibrar a exploração de boas soluções e a exploração do espaço de busca. As regras de transição são apresentadas a seguir.

$$p^k_{ij} = \begin{cases} \frac{(\tau^k_{ij})^\alpha (\eta^k_{ij})^\beta (\sigma_{ij})^\gamma}{\sum_{q=1}^{2} \sum_{l \in N(i)} (\tau^q_{il})^\alpha (\eta^q_{il})^\beta (\sigma_{il})^\gamma}, & se\ j \in N(i) \\ 0 \quad , \quad cc. \end{cases}, \forall k = 1,2 \tag{43}$$

Em (43), $\eta^k_{ij}$ denota a visibilidade do arco $i-j$, com $k=1$, $\eta^1_{ij} = b_j/(c_{ij}+c_{j1})$ denota a visibilidade do arco $i-j$ com o bônus no vértice $j$ sendo coletado, e com $k=2$, $\eta^2_{ij} = 1/(c_{ij}+c_{j1})$ expressa a visibilidade do arco $i-j$ sem

que o bônus no vértice $j$ seja coletado;$\sigma_{ij}$ representa a visibilidade dos passageiros que trafegam de $i$ para $j$ que é definido pela quantidade de passageiros que podem ser embarcados com origem $i$ e destino $j$; e $\alpha$, $\beta$, $\gamma$ são parâmetros que determinam as influências relativas da trilha do feromônio, do valor de visibilidade do arco e o valor da informação heurística dos passageiros, respectivamente. $N(i)$ representa o conjunto de localidades não visitadas que podem ser alcançadas a partir de $i$. A próxima localidade a ser inserida na rota da formiga é determinada por seleção estocástica baseada nas probabilidades de transição apresentadas na equação (43). Caso o sorteio tenha selecionado uma opção da equação (43) com $k = 1$ o vértice $j$ será inserido na rota com o bônus coletado e caso seja pelo $k = 2$ o vértice $j$ será inserido na rota sem que o bônus seja coletado. Na medida em que as inserções são realizadas, a viabilização do passageiros, desempenhada por embarPass(), é executada junto ao incremento da variável *numCh* e a execução de testeCh() para verificar a quantidade de chamadas da função objetivo realizadas até o momento. O método de construção retorna a menor solução encontrada no decorrer das inserções.

Após todas as formigas construírem sua solução, a melhor formiga $S^*$ recebe a busca VND, conforme passo 20, na tentativa de refinar a solução. Logo em seguida, a atualização dos feromônios é realizada pelo método atualizarFero($\tau$, $\rho$, *pesoSol*, $S$, $S^*$) que atualiza a tabela de feromônios, valores $\tau$; $\rho$ a taxa de evaporação e $\omega$ um peso que irá modular a contribuição da solução global $S$ e da local $S^*$ são usados no processo. A atualização dos feromônios é realizada da seguinte maneira:

$$\tau_{ij}^k = (1-\rho)\cdot\tau_{ij}^k + \rho\cdot\Delta_{ij}^k,\ \forall k = 1,2 \tag{44}$$

Onde,

$$\Delta_{ij}^k = \begin{cases} \omega\cdot\frac{2}{f(S)+F} + (1-\omega)\cdot\frac{2}{f(S)+F} & ,\ caso\,1 \\ \omega\cdot\frac{1}{f(S)+F} + (1-\omega)\cdot\frac{1}{f(S)+F} & ,\ caso\,2 \\ 0 & ,\ cc. \end{cases} \quad \forall k = 1,2 \tag{45}$$

Em 45, $F$ é uma constante grande utilizada para garantir valores positivos nas soluções. O caso 1 ocorre quando a aresta $(i,j)$ com a opção de coleta $k$ na localidade $j$ acontece e pertence a ambas soluções ($S$ e $S^*$). O caso 2 acontece somente quando a aresta $(i,j)$ com a opção de coleta $k$ no vértice $j$ acontece e pertence a uma das soluções ($S$ ou $S^*$). A opção de coleta $k = 1$ acontece quando o bônus é coletado na localidade $j$ e $k = 2$ quando não ocorre a coleta. Após a atualização dos feromônios, uma nova iteração do algoritmo inicia. O algoritmo ACO encerra quando atinge o número de chamadas a função objetivo máximo, *maxFc*, ou quando a melhor solução não é melhorada em *maxConv* iterações. A melhor solução é retornada quando um dos critérios de parada são satisfeitos.

## 6. Banco de Instâncias

Como o PCVP-BoTc é um modelo inovador, a literatura não disponibiliza instâncias *benchmark* para a avaliação da eficiência computacional dos modelos de programação sugeridos ou a performance computacional dos algoritmos propostos. Dessa forma, foi criado um banco de instâncias, denominado PCVP-BoTcLIB para viabilizar a realização de testes. O projeto dos casos de teste do banco considerou vários fatores que podem afetar o comportamento da solução do problema. Os fatores incluem: a capacidade do veículo do caixeiro, os valores dos lucros associado a cada localidade, o tempo de coleta dos bônus, características da distribuição de passageiros sobre as localidades, número de passageiros disponíveis por vértice, o valor das restrições de tarifa e de tempo máximo e, finalmente o intervalo das janelas de tempo.

As distâncias entre localidades são contadas em uma matriz simétrica. Foram construídos casos testes com o número de localidades variando segundo o conjunto $V$ = (5, 6, ..., 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500); O número de passageiros $P = 2\cdot|V|$ e a capacidade do veículo $C = (3,4)$. As arestas de custo e tempo foram sorteadas no intervalo [100, 250]. O nível de satisfação (lucro), associado a cada vértice foi definido por $b_i = \lfloor 100\cdot\frac{c_{1i}}{\theta}\rfloor$, onde $\theta = \max_{w\in V} c_{1w}$, porém 15 a 20% dos vértices tiveram seus lucros modificados para valores sorteados entre [150, 200]. Os tempos de coleta dos bônus foram sorteados em [10, 60]. A localidade inicial possui bônus e tempo de serviço iguais a 0.

Cada localidade recebeu dois passageiros. Conhecida a origem do passageiro, seu destino é sorteado aleatoriamente. A despesa máxima admissível e o tempo máximo de permanência para o passageiro foram calculados com base no caminho mais curto de Dijkstra [12]. O caminho mínimo entre a origem e o destino de cada passageiro foi calculado sobre as arestas de custo e tempo. Seja *Lc* o caminho mínimo sobre as arestas de custo e *Lt* o caminho mínimo sobre as arestas de tempo entre os pontos de embarque e desembarque do passageiro $p$. A despesa máxima do passageiro $p$ é dada pelo custo de *Lc* multiplicado por um fator definido uniformemente entre [2,4]. De maneira análoga, o tempo máximo de permanência embarcado do passageiro $p$ é dado pelo custo de *Lt* multiplicado também por um fator selecionado uniformemente entre [2,4].

Por fim, as janelas de tempo para cada passageiro foram escolhidas aleatoriamente dentre as três apresentadas na Tabela 1. A coluna *Num* apresenta os números que cada janela de tempo representa. A coluna *JanelaIF* denota o início e fim para cada janela de tempo. O valor de *arestaM* é dado pela média das arestas sobre a matriz de tempo.

As instâncias criadas formam o conjunto *benchmark* para o problema e está disponível em http://www.dimap.ufrn.br/lae/downloads/PCVP-BoTcLIB.tar.gz.

**Table 1.** Janelas de Tempo

| Num | JanelaIF |
|---|---|
| 1 | $[0, arestaM \cdot \|V\| \cdot 0.33]$ |
| 2 | $[arestaM \cdot \|V\| \cdot 0.33, arestaM \cdot \|V\| \cdot 0.66]$ |
| 3 | $[arestaM \cdot \|V\| \cdot 0.66, arestaM \cdot \|V\| \cdot 0.99]$ |

## 7. Experimentos Computacionais

Com a finalidade de validar as formulações quadráticas e os algoritmos propostos, os mesmos foram implementados em um PC Intel Core i7, 4G de RAM, rodando Ubuntu 16.04.2 LTS e utilizando a linguagem C++. O Gurobi 6.5.2 [23] foi utilizado como *solver*.

Esta seção é dividida em três subseções. A subseção 7.1 apresenta os resultados exatos das formulações matemáticas. Na seção 7.2 é feita uma comparação entre os resultados das formulações com a melhor solução dos algoritmos propostos. Os resultados para os algoritmos propostos sobre o banco de instâncias são apresentados na seção 7.3.

O solver foi limitado a 80000 segundos de processamento. Os algoritmos GVND1, GNVD2 e ACO foram testados em trinta execuções independentes. O algoritmo NVH só é executado uma vez por ter suas duas fases solucionadas de forma exata através do *solver*.

O algoritmo NVH recebeu o valor de 10000s para os parâmetros *tempoRota* e *tempoPass*. Os algoritmos GVND1, GNVD2 e ACO foram parametrizados através do uso da ferramenta *irace* [36]. Um grupo de instâncias sorteadas da biblioteca PCVP-BoTcLIB serviu como base para o *irace*, As instâncias usadas na configuração dos parâmetros não fizeram parte das instâncias do experimento computacional. No *irace*, cada parâmetro tem um domínio associado a seu respectivo tipo (inteiro, real, categórico ou ordinal). Os parâmetros utilizados foram do tipo inteiro e real. Nove parâmetros foram fixados pelo *irace*. A Tabela 2 apresenta a descrição dos parâmetros. A Tabela 3 mostra as configurações sugeridas pelo *irace* para os parâmetros. O símbolo "-" significa que esse parâmetro não foi utilizado no algoritmo em questão.

**Table 2.** Parâmetros usados pelo *irace*

| Parâmetro | Intervalo | Tipo | ALGORITMO |
|---|---|---|---|
| *tamLista* | [0.1, 1] | Real | GVND1 e 2 |
| *maxFc* | [1000, 150000] | Inteiro | GVND1 e 2, ACO |
| *maxConv* | [10, 300] | Inteiro | GVND1 e 2, ACO |
| $\rho$ | [0.01, 1.0] | Real | ACO |
| $\alpha$ | [1.0, 2.0] | Real | ACO |
| $\beta$ | [2.0, 3.0] | Real | ACO |
| $\gamma$ | [3.0, 4.0] | Real | ACO |
| *numF* | [1, 10] | Inteiro | ACO |
| $\omega$ | [0.1, 1.0] | Real | ACO |

A análise do desempenho dos algoritmos usou como métricas o melhor valor obtido, a média dos valores e o tempo médio obtidos nas rodadas e em cada instância. O teste de Friedman [18] foi usado para verificar diferenças significativas. O nível de significância adotado foi $\alpha = 0,05$. Uma vez que o teste de Friedman rejeitou a hipótese nula, o teste *post-hoc* de Siegel e Castellan [42] foi realizado para encontrar as comparações entre os pares de algoritmos.

**Table 3.** Parâmetros dos algoritmos

| Parâmetro | GVND1 | GVND2 | ACO |
|---|---|---|---|
| *tamLista* | 0.1146 | 0.9325 | - |
| *maxFc* | 148061 | 84378 | 72810 |
| *maxConv* | 94 | 206 | 63 |
| $\rho$ | - | - | 0.1238 |
| $\alpha$ | - | - | 1.1635 |
| $\beta$ | - | - | 2.0104 |
| $\gamma$ | - | - | 3.4607 |
| *numF* | - | - | 2 |
| $\omega$ | - | - | 0.3401 |

### 7.1 Resultados Exatos

Os resultados computacionais mostram soluções obtidas pela aplicação do solver a 28 instâncias da PCVP-BoTcLIB com até 40 cidades, para as duas formulações. A Tabela 4 mostra os resultados. As colunas *Nome*, $|V|$, $|P|$ e $C$ estão relacionadas com as características da instância e são, respectivamente, o nome da instância, o número de vértices, e a capacidade do veículo. As colunas *Sol*, *LB*, *GAP* e *T(s)* para ambos os modelos *FSQ* e *MFQ*, estão relacionadas com a solução obtida pelo solver, e mostram o valor da solução obtida, o limite inferior, o desvio percentual do valor mostrado na coluna *Sol* do limite inferior (*LB*) calculado pelo solucionador, e o tempo de processamento em segundos.

O valor "0" na coluna *GAP* indica que a instância foi resolvida de forma ótima com o modelo correspondente. O valor "80000" na coluna *T(s)* indica que o solver parou com o esgotamento do tempo de processamento.

Os resultados apresentados na Tabela 4 mostram que o solver conseguiu encontrar a solução ótima de 14 instâncias para o modelo *FSQ* e 13 instâncias para o modelo *MFQ*. Os problemas que ficaram sem garantia de solução ótima para ambos os modelos foram similares, com exceção da instância "*n11c3*" que o modelo *FSQ* conseguiu solucionar. A parada se deu pela limitação de tempo disponível para a solução do problema. Os tempos de processamento do modelo *FSQ* foram sistematicamente menores que os gastos pelo modelo *MFQ*. A diferença fica evidenciada em alguns problemas, a exemplo do problema "*n10c4*", onde o modelo *FSQ* teve um tempo de processamento de 1595.37 segundos para alcançar o ótimo, enquanto que para o modelo *MFQ* o tempo foi de 5360.16 segundos. A diferença é confirmada nos desvios percentuais alcançados em ambos os modelos. Os valores de *GAP* são maiores para o modelo *MFQ*.

Para o conjunto de instâncias examinadas, verifica-se que o modelo *FSQ* possui uma eficiência superior quando comparado ao modelo *MFQ*.

**Table 4.** Resultados obtidos pelo solver para as formulações quadráticas.

| Instância | | | | FSQ | | | | MFQ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Nome | $\|V\|$ | $\|P\|$ | $C$ | Sol | LB | GAP | T(s) | Sol | LB | GAP | T(s) |
| n5c3 | 5 | 10 | 3 | 0 | 0 | 0 | 2.1789 | 0 | 0 | 0 | 0,7588 |
| n5c4 | 5 | 10 | 3 | 0 | 0 | 0 | 0.3550 | 0 | 0 | 0 | 0,5628 |
| n6c3 | 6 | 12 | 4 | -39 | -39 | 0 | 1.0086 | -39 | -39 | 0 | 1,7308 |
| n6c4 | 6 | 12 | 4 | -2 | -2 | 0 | 8.0479 | -2 | -2 | 0 | 10,8071 |
| n7c3 | 7 | 14 | 3 | -33,33 | -33,33 | 0 | 14.4717 | -33 | -33 | 0 | 23,5006 |
| n7c4 | 7 | 14 | 3 | -96 | -96 | 0 | 12.5429 | -96 | -96 | 0 | 45,875 |
| n8c3 | 8 | 16 | 4 | -67,25 | -67,25 | 0 | 75.8871 | -67,25 | -67.25 | 0 | 176,018 |
| n8c4 | 8 | 16 | 4 | 0 | 0 | 0 | 104.708 | 0 | 0 | 0 | 132,1 |
| n9c3 | 9 | 18 | 3 | -260,917 | -260,917 | 0 | 606.145 | -260,917 | -260,917 | 0 | 1319,85 |
| n9c4 | 9 | 18 | 4 | -257,167 | -257,167 | 0 | 1094.96 | -257,167 | -257,160 | 0 | 1206,25 |
| n10c3 | 10 | 20 | 3 | -88,833 | -88,8333 | 0 | 3379 | -88,8333 | -88,8333 | 0 | 5908,48 |
| n10c4 | 10 | 20 | 4 | -243,667 | -243,667 | 0 | 1595.37 | -243,667 | -243,667 | 0 | 5360,16 |
| n11c3 | 11 | 22 | 3 | -358,833 | -358,833 | 0 | 40030.1 | -358,833 | -651,365 | 0,8152 | 80000 |
| n11c4 | 11 | 22 | 4 | -163,083 | -163,083 | 0 | 18268.7 | -163,083 | -163,083 | 0 | 41931,1 |
| n12c3 | 12 | 24 | 3 | -202,5 | -470,249 | 1,32 | 80000 | -187,167 | -627,595 | 2,3531 | 80000 |
| n12c4 | 12 | 24 | 4 | -264,167 | -494,771 | 0,87 | 80000 | -259,833 | -762,299 | 1,9338 | 80000 |
| n13c3 | 13 | 26 | 3 | -178,25 | -669,959 | 2,75 | 80000 | -137,333 | -716,017 | 4,21 | 80000 |
| n13c4 | 13 | 26 | 4 | -319,25 | -688,452 | 1,15 | 80000 | -275,917 | -806,418 | 1,92 | 80000 |
| n14c3 | 14 | 28 | 3 | -303 | -945,994 | 2,12 | 80000 | -219 | -997,351 | 3,55 | 80000 |
| n14c4 | 14 | 28 | 4 | -298,5 | -970,675 | 2,25 | 80000 | -229,667 | -1016,57 | 3,42 | 80000 |
| n15c3 | 15 | 30 | 3 | -262,667 | -980,814 | 2,73 | 80000 | -109,667 | -1022,28 | 8,32 | 80000 |
| n15c4 | 15 | 30 | 4 | -197,677 | -1197,87 | 5,06 | 80000 | -126,833 | -1178,59 | 8,29 | 80000 |
| n20c3 | 20 | 40 | 3 | 0 | -1665 | 1e+100 | 80000 | 0 | -1563,12 | 1e+100 | 80000 |
| n20c4 | 20 | 40 | 4 | -264,58 | -1487,5 | 4,622 | 80000 | 0 | -1484,46 | 1e+100 | 80000 |
| n30c3 | 30 | 60 | 3 | 0 | -2459,2 | 1e+100 | 80000 | 0 | -2460 | 1e+100 | 80000 |
| n30c4 | 30 | 60 | 4 | 0 | -2535 | 1e+100 | 80000 | 0 | -2542 | 1e+100 | 80000 |
| n40c3 | 40 | 80 | 3 | 0 | -3380 | 1e+100 | 80000 | 0 | -3380 | 1e+100 | 80000 |
| n40c4 | 40 | 80 | 4 | 0 | -3292 | 1e+100 | 80000 | 0 | -3292 | 1e+100 | 80000 |

### 7.2 Comparação entre Resultados Exatos e Heurísticos

A Tabela 5 resume os melhores resultados alcançados pela tentativa de solução exata dos modelos do problema ao lado das melhores soluções encontradas pelos vários algoritmos desenvolvidos. As colunas *Nome*, *Sol*, *LB* exercem a mesma função da Tabela 4. As colunas *Min* e *T(s)*, mostram, respectivamente, o valor da melhor solução e o tempo de processamento médio em segundos.

A tabela mostra que os resultados do algoritmo NVH ficam distantes dos valores ótimos e de alguns limites obtidos pelo *solver*. Os algoritmos GVND1, GVND2 e ACO conseguiram encontrar o ótimo para a maioria das instâncias que possuem resultado exato, porém os algoritmos GVND1 e ACO foram os que obtiveram maior sucesso. Os algoritmos GVND1, GVND2 e ACO apresentaram bom desempenho mesmo nas soluções em que o *solver* falha em obter a solução ótima, em virtude de suas soluções estarem contidas nos limites encontrados.

O tempo de processamento dos algoritmos heurísticos é significativamente menor que o empregado pelo *solver*. Nas instâncias com mais de 12 cidades, o *solver* praticamente usou o tempo de processamento máximo de 22 horas enquanto que os algoritmos heurísticos encontraram soluções melhores em segundos.

### 7.3 Resultados Heurísticos

A Tabela 6 resume os resultados qualitativos. 48 instâncias foram examinadas. As colunas *Nome*, $|V|$, $|P|$ e $C$ desempenham a mesma função da Tabela 4. As funções das colunas *Min* e *T(s)* foram apresentadas na Tabela 5. A coluna *Méd.* representa o valor médio das soluções mínimas nas 30 execuções. O teste estatístico de Friedman, é aplicado para avaliar o desempenho de cada algoritmo. As colunas da Tabela 6 reportam o valor mínimo encontrado (*Min*), a média dos valores encontrados (*Méd.*) e o tempo médio de processamento (*T(s)*). O teste de Friedman constatou diferenças significativas em todas os indicadores qualitativos, com *p-valores* menores que 0,001. Então, o *post-hoc* de Siegel e Castellan foi aplicado para comparar os pares de algoritmos sobre as variáveis. As tabelas 7 e 8 mostram os *p-valores* encontrados da análise pareada de algoritmos nas colunas *Méd* e *T(s)*.

No contexto da solução média, diferenças significativas são mostradas na Tabela 7 entre o algoritmo NVH com os

**Table 5.** Resultados do solver e dos algoritmos propostos.

| Instância | Solver | | NVH | | GVND1 | | GVND2 | | ACO | |
|---|---|---|---|---|---|---|---|---|---|---|
| Nome | Sol | LB | Min | T(s) | Min | T(s) | Min | T(s) | Min | T(s) |
| n5c3 | 0 | 0 | 0 | 0 | **0** | 0,90 | **0** | 1,03 | **0** | 0,50 |
| n5c4 | 0 | 0 | 0 | 0 | **0** | 8,40 | **0** | 1,00 | **0** | 0,76 |
| n6c3 | -39 | -39 | 0 | 0 | **-39,00** | 26,47 | 0 | 1,27 | **-39,00** | 5,03 |
| n6c4 | -2 | -2 | 0 | 0 | 0 | 18,53 | 0 | 1,33 | **-2,00** | 0,93 |
| n7c3 | -33,33 | -33,33 | 0 | 0 | 0 | 2,50 | 0 | 1,30 | **-33,33** | 1,10 |
| n7c4 | -96 | -96 | 0 | 0 | **-96,00** | 37,63 | **-96,00** | 4,40 | **-96,00** | 7,66 |
| n8c3 | -67,25 | -67,25 | 0 | 0 | 0 | 2,67 | 0 | 1,53 | **-67,25** | 3,07 |
| n8c4 | 0 | 0 | 0 | 0 | **0** | 1,67 | **0** | 1,70 | **0** | 0,57 |
| n9c3 | -260,917 | -260,917 | 0 | 0 | **-260,92** | 40,17 | **-260,92** | 44,97 | **-260,92** | 12,56 |
| n9c4 | -257,167 | -257,167 | 0 | 0 | **-257,17** | 47,33 | **-257,17** | 22,17 | **-257,17** | 20,63 |
| n10c3 | -88,833 | -88,8333 | 0 | 0 | -85,83 | 14,67 | -71,75 | 12,60 | -71,75 | 10,73 |
| n10c4 | -243,667 | -243,667 | 0 | 0 | **-243,67** | 36,33 | -226,17 | 15,10 | **-243,67** | 32,73 |
| n11c3 | -358,833 | -358,833 | 0 | 0 | -317,17 | 40,17 | -358,83 | 8,63 | -317,17 | 54,50 |
| n11c4 | -163,083 | -163,083 | 0 | 0 | -117,67 | 9,80 | -90,33 | 3,83 | **-163,08** | 9,60 |
| n12c3 | -202,5 | -470,249 | 0 | 0 | -58,67 | 29,53 | 0 | 2,33 | -115,33 | 3,77 |
| n12c4 | -264,167 | -494,771 | 0 | 0 | -183,75 | 6,40 | -226,33 | 8,10 | -167,50 | 3,80 |
| n13c3 | -178,25 | -669,959 | 0 | 0 | 0 | 7,07 | 0 | 2,53 | -164,17 | 9,13 |
| n13c4 | -319,25 | -688,452 | 0 | 0 | -279,83 | 34,37 | 0 | 2,27 | -102,83 | 2,20 |
| n14c3 | -303 | -945,994 | 0 | 0 | -383,92 | 101,27 | -312,67 | 21,43 | -337,58 | 13,23 |
| n14c4 | -298,5 | -970,675 | 0 | 0 | -323,75 | 13,77 | -199,00 | 16,73 | -316,42 | 34,20 |
| n15c3 | -262,667 | -980,814 | 0 | 0 | -327,33 | 66,97 | -334,58 | 40,37 | -332,58 | 17,47 |
| n15c4 | -197,677 | -1197,87 | 0 | 0 | -298,43 | 29,87 | -225,67 | 14,50 | -225,67 | 14,53 |
| n20c3 | 0 | -1665 | -78 | 1 | -528,42 | 35,73 | -511,08 | 7,97 | -566,17 | 56,83 |
| n20c4 | -264,58 | -1487,5 | -2 | 0 | -369,17 | 39 | -439,08 | 14,00 | -448,72 | 40,9 |
| n30c3 | 0 | -2459,2 | -72 | 1 | -402,33 | 25,3 | -153,50 | 8,43 | -338 | 18,17 |
| n30c4 | 0 | -2535 | -118 | 1 | -638,67 | 27,67 | -893,10 | 31,73 | -970,67 | 92,03 |
| n40c3 | 0 | -3380 | -202 | 4 | -711 | 118,13 | -635,50 | 65,03 | -697,17 | 135,8 |
| n40c4 | 0 | -3292 | -200 | 1 | -682,83 | 79,87 | -308,83 | 10,80 | -682,83 | 111,53 |

outros métodos heurísticos e do algoritmo GVND2 com o GVND1 e ACO, dado que os *p-valores* correspondentes são menores que o nível de significância. O algoritmo NVH obteve pior desempenho na busca de soluções em comparação aos demais. A Tabela 7 mostra que também houve diferenças do algoritmo GVND2 com os GVND1 e ACO.

O algoritmo GVND1 encontrou melhores valores mínimos e médios, respectivamente em 29 e 31 instâncias, enquanto o GVND2 encontrou melhores valores para 9 e 7 dos casos de teste. O algoritmo ACO comparado com o GVND2 encontrou melhores valores sobre 31 e 32 casos de teste, enquanto que o GVND2 encontrou melhores valores sobre 9 e 11 instâncias. Estes resultados indicam que os algoritmos GVND1 e ACO são os melhores nas instâncias referindo ao indicador de solução.

A Tabela 8 mostra os *p-valores* para a variável de tempo médio de processamento. Os resultados mostraram que ocorreram diferenças significativas entre quase todos os algoritmos, menos entre o GVND2 e ACO. Comparando o NVH aos outros, o NVH teve o menor custo computacional em comparação com os demais em instâncias de até 100 cidades. Para instâncias maiores que 100 cidades o NVH perde conclusivamente eficiência de processamento em tempo. Na comparação entre os algoritmos GVND1, GVND2 e ACO os resultados mostram que o GVND1 comparado com o GVND2 obteve melhores tempos somente em 7 instâncias. GVND2 obteve melhores tempo em 41 casos de testes. No caso que o GVND1 é comparado com o algoritmo ACO, o GVND1 obteve melhores tempo em 17 instâncias, enquanto o ACO em 31 instâncias. Estes dados confirmam que o NVH é eficiente somente para as instâncias abaixo de 100 localidades e que o GVND2 e ACO foram os algoritmos com os melhores tempos.

A partir dos resultados, é possível concluir que a técnica utilizada pelo NVH de dividir a solução do problema em duas etapas de solução exata falha na solução de problemas de porte maior, tanto em eficiência computacional quanto na qualidade da solução alcançada. Os algoritmos GVND1 e ACO foram os que obtiveram os melhores resultados em relação à qualidade de solução, GVND1 demandando mais esforço computacional. Os algoritmos GVND2 e ACO foram os que obtiveram os melhores tempos de processamento, porém o GVND2 não alcançando tão boas soluções quanto GVND1 e ACO. Finalmente, pode-se concluir que dentre todas as metaheurísticas o algoritmo ACO apresentou os resultados mais promissores, alcançando boa qualidade de solução e tempo

de processamento similares ao GVND2.

## 8. Conclusões

Este artigo apresentou o Problema do Caixeiro Viajante com Coleta Opcional de Bônus, Tempo de Coleta e Passageiros (PCVP-BoTc), uma nova generalização para o clássico Problema do Caixeiro Viajante (PCV). O PCVP-BoTc pode representar um modelo de otimização importante para tratar situações típicas de transporte colaborativo e, em especial, quando serviços de *courier* estão envolvidos. Foram propostas estratégias exatas e heurísticas para a resolução do problema. Com objetivo de validar a efetividade das abordagens, foi elaborado um banco de instâncias denominado de PCVP-BoTcLIB que considera, de forma estruturada, vários fatores que podem afetar o comportamento da solução do problema.

Foram elaboradas duas formulações matemáticas para o problema. A eficiência das formulações foi examinada através do solver Gurobi. Um experimento baseado em 28 instâncias mostrou que 14 instâncias foram resolvidas de forma ótima pelo modelo *FSQ* e, para as 14 instâncias restante, o modelo permitiu o cálculo de limites de solução. Os resultados do modelo *FSQ* se mostraram superiores ao modelo *MFQ*. Foram analisados quatro algoritmos heurísticos para a solução do problema, um algoritmo com duas fases exatas solucionado através de Programação Linear e três meta-heurísticas hibridizadas. Os algoritmos meta-heurísticos analisadas foram hibridizados com procedimentos de busca usando técnicas do tipo VND (Variable Neighbourhood. Descent) e métodos de PL. Foram criados operadores de vizinhança para o VND específicos para o problema e outros adaptados da literatura. Um novo método de construção de soluções foi proposto e utilizado no algoritmo GVND2 e no ACO. Um experimento sobre 48 instâncias foi realizado para verificar a eficiência dos algoritmos elaborados. No conjunto de instâncias examinadas, o melhor desempenho em qualidade de solução foi alcançado por algoritmos meta-heurísticos. O algoritmo NVH é conclusivamente ineficiente na solução das maiores instâncias do conjunto examinado. Os resultados do método NVH demonstraram que a técnica de separar a decisão do embarque da determinação da melhor rota com coleta de bônus é ineficaz. O teste estatístico visando detectar diferenças significativas no comportamento dos algoritmos meta-heurísticos em termos de solução e tempo médio de processamento, apresentou resultados que favorecem o algoritmo ACO, indicando-o como uma alternativa promissora para a solução do problema na maioria das instâncias. O problema desta pesquisa possui uma grande dificuldade de solução pois acopla quatro variáveis de decisão: A escolha das localidades da rota, a opção de coletar o bônus nas localidades visitadas, a rota entra as cidades e a decisão de embarque dos passageiros. Para atender as quatro decisões acopladas a busca local baseada em VND examinou vizinhanças associadas a cada uma dessas decisões.

Em virtude da característica inovadora do modelo examinado, diversas questões de pesquisas podem ser investigadas em futuros trabalhos como: o desempenho de outras meta-heurísticas e a vantagem de tratar os vários interesses conflitantes entre passageiros e o motorista como o tempo de viagem e a tarifa máxima admissível através de formulação multiobjetivo.

## Agradecimentos

Este trabalho foi suportado pelo CNPq através dos processos 302387/2016-1 e 306702/2017-7.

## Contribuição dos autores

Os autores contribuíram de forma equivalente para o desenvolvimento do trabalho.

## Referências

[1] Agatz, N., Erera, A., Savelsbergh, M., and Wang, X. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research 223*, 2 (2012), 295–303.

[2] Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., and Rus, D. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences 114*, 3 (2017), 462–467.

[3] Archetti, C., Feillet, D., Hertz, A., and Speranza, M. G. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society 60*, 6 (2009), 831–842.

[4] Archetti, C., Savelsbergh, M., and Speranza, M. G. The vehicle routing problem with occasional drivers. *European Journal of Operational Research 254*, 2 (2016), 472–480.

[5] Arslan, A. M., Agatz, N., Kroon, L., and Zuidwijk, R. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science 53*, 1 (2018), 222–235.

[6] Balas, E. The prize collecting traveling salesman problem. *Networks 19*, 6 (1989), 621–636.

[7] Berbeglia, G., Cordeau, J.-F., and Laporte, G. Dynamic pickup and delivery problems. *European journal of operational research 202*, 1 (2010), 8–15.

[8] Calheiros, Z. S. A. O problema do caixeiro viajante com passageiros. Master's thesis, Brasil, 2017.

[9] Claus, A. A new formulation for the travelling salesman problem. *SIAM Journal on Algebraic Discrete Methods 5*, 1 (1984), 21–25.

[10] Cordeau, J.-F., and Laporte, G. The dial-a-ride problem: models and algorithms. *Annals of operations research 153*, 1 (2007), 29–46.

**Table 6.** Resultados obtidos pelos algoritmos propostos ao problema.

| Instância | | | | NVH | | | GVND1 | | | GVND2 | | | ACO | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nome | $\|V\|$ | $\|P\|$ | $C$ | Min | Méd. | T(s) | Min | Méd. | T(s) | Min | Méd. | T(s) | Min | Méd. | T(s) |
| n5c3 | 5 | 10 | 3 | 0 | 0 | 0 | 0 | 0 | 0,90 | 0 | 0 | 1,03 | 0 | 0 | 0,50 |
| n5c4 | 5 | 10 | 3 | 0 | 0 | 0 | 0 | 0 | 8,40 | 0 | 0 | 1,00 | 0 | 0 | 0,76 |
| n6c3 | 6 | 12 | 4 | 0 | 0 | 0 | -39,00 | -39,00 | 26,47 | 0 | 0 | 1,27 | -39,00 | -39,00 | 5,03 |
| n6c4 | 6 | 12 | 4 | 0 | 0 | 0 | 0 | 0 | 18,53 | 0 | 0 | 1,33 | -2,00 | -1,40 | 0,93 |
| n7c3 | 7 | 14 | 3 | 0 | 0 | 0 | 0 | 0 | 2,50 | 0 | 0 | 1,30 | -33,33 | -24,44 | 1,10 |
| n7c4 | 7 | 14 | 3 | 0 | 0 | 0 | -96,00 | -96,00 | 37,63 | -96,00 | -96,00 | 4,40 | -96,00 | -96,00 | 7,66 |
| n8c3 | 8 | 16 | 4 | 0 | 0 | 0 | 0 | 0 | 2,67 | 0 | 0 | 1,53 | -67,25 | -41,79 | 3,07 |
| n8c4 | 8 | 16 | 4 | 0 | 0 | 0 | 0 | 0 | 1,67 | 0 | 0 | 1,70 | 0 | 0 | 0,57 |
| n9c3 | 9 | 18 | 3 | 0 | 0 | 0 | -260,92 | -260,92 | 40,17 | -260,92 | -260,92 | 44,97 | -260,92 | -255,85 | 12,56 |
| n9c4 | 9 | 18 | 4 | 0 | 0 | 0 | -257,17 | -257,17 | 47,33 | -257,17 | -257,17 | 22,17 | -257,17 | -257,17 | 20,63 |
| n10c3 | 10 | 20 | 3 | 0 | 0 | 0 | -85,83 | -68,22 | 14,67 | -71,75 | -71,75 | 12,60 | -71,75 | -52,73 | 10,73 |
| n10c4 | 10 | 20 | 4 | 0 | 0 | 0 | -243,67 | -243,67 | 36,33 | -226,17 | -226,17 | 15,10 | -243,67 | -240,78 | 32,73 |
| n11c3 | 11 | 22 | 3 | 0 | 0 | 0 | -317,17 | -317,17 | 40,17 | -358,83 | -294,06 | 8,63 | -317,17 | -305,77 | 54,50 |
| n11c4 | 11 | 22 | 4 | 0 | 0 | 0 | -117,67 | -117,67 | 9,80 | -90,33 | -87,90 | 3,83 | -163,08 | -115,20 | 9,60 |
| n12c3 | 12 | 24 | 3 | 0 | 0 | 0 | -58,67 | -50,33 | 29,53 | 0 | 0 | 2,33 | -115,33 | -53,78 | 3,77 |
| n12c4 | 12 | 24 | 4 | 0 | 0 | 0 | -183,75 | -85,47 | 6,40 | -226,33 | -188,03 | 8,10 | -167,50 | -141,32 | 3,80 |
| n13c3 | 13 | 26 | 3 | 0 | 0 | 0 | 0 | 0 | 7,07 | 0 | 0 | 2,53 | -164,17 | -88,66 | 9,13 |
| n13c4 | 13 | 26 | 4 | 0 | 0 | 0 | -279,83 | -248,47 | 34,37 | 0 | 0 | 2,27 | -102,83 | -11,63 | 2,20 |
| n14c3 | 14 | 28 | 3 | 0 | 0 | 0 | -383,92 | -346,86 | 101,27 | -312,67 | -236,82 | 21,43 | -337,58 | -213,04 | 13,23 |
| n14c4 | 14 | 28 | 4 | 0 | 0 | 0 | -323,75 | -206,93 | 13,77 | -199,00 | -196,67 | 16,73 | -316,42 | -262,74 | 34,20 |
| n15c3 | 15 | 30 | 3 | 0 | 0 | 0 | -327,33 | -306,68 | 66,97 | -334,58 | -219,71 | 40,37 | -332,58 | -227,40 | 17,47 |
| n15c4 | 15 | 30 | 4 | 0 | 0 | 0 | -298,43 | -293,41 | 29,87 | -225,67 | -225,67 | 14,50 | -225,67 | -218,57 | 14,53 |
| n20c3 | 20 | 40 | 3 | -78 | -78 | 1 | -528,42 | -357,48 | 35,73 | -511,08 | -125,09 | 7,97 | -566,17 | -394,8 | 56,83 |
| n20c4 | 20 | 40 | 4 | -2 | -2 | 0 | -369,17 | -339,6 | 39 | -439,08 | -276,83 | 14,00 | -448,72 | -282,31 | 40,9 |
| n30c3 | 30 | 60 | 3 | -72 | -72 | 1 | -402,33 | -209,67 | 25,3 | -153,50 | -20,37 | 8,43 | -338 | -192,3 | 18,17 |
| n30c4 | 30 | 60 | 4 | -118 | -118 | 1 | -638,67 | -462,7 | 27,67 | -893,10 | -629,88 | 31,73 | -970,67 | -626,94 | 92,03 |
| n40c3 | 40 | 80 | 3 | -202 | -202 | 4 | -711 | -643,52 | 118,13 | -635,50 | -589,74 | 65,03 | -697,17 | -620,89 | 135,8 |
| n40c4 | 40 | 80 | 4 | -200 | -200 | 1 | -682,83 | -625,74 | 79,87 | -308,83 | -10,29 | 10,80 | -682,83 | -593,41 | 111,53 |
| n50c3 | 50 | 100 | 3 | -319 | -319 | 1 | -789,17 | -555,83 | 46,67 | -647,00 | -343,07 | 31,10 | -813 | -641,92 | 62,53 |
| n50c4 | 50 | 100 | 4 | -380,5 | -380,5 | 4 | -698,83 | -563,34 | 94,7 | -694,83 | -648,54 | 146,10 | -764,5 | -559,98 | 81,5 |
| n60c3 | 60 | 120 | 3 | -382 | -382 | 4 | -899,17 | -763,23 | 105,57 | -408,50 | -13,62 | 21,77 | -909 | -849,19 | 145,6 |
| n60c4 | 60 | 120 | 4 | -718,5 | -718,5 | 9 | -1355,5 | -1179,69 | 242,97 | -1280,17 | -1153,29 | 149,07 | -1369,67 | -1197,49 | 261,33 |
| n70c3 | 70 | 140 | 3 | -689 | -689 | 6 | -1271 | -1025,43 | 235,6 | -1240,67 | -1109,01 | 171,43 | -1220,17 | -1066,99 | 159,63 |
| n70c4 | 70 | 140 | 4 | -612,5 | -612,5 | 7 | -1138,5 | -996,32 | 195,5 | -1001,00 | -928,27 | 174,97 | -1250,17 | -1026,59 | 139,97 |
| n80c3 | 80 | 160 | 3 | -679 | -679 | 10 | -1423,67 | -906,54 | 167,27 | -685,00 | -508,23 | 61,57 | -1334,5 | -1040,22 | 257,7 |
| n80c4 | 80 | 160 | 4 | -491 | -491 | 24 | -903,67 | -751,32 | 97,5 | -693,50 | -566,68 | 63,97 | -1053,17 | -857,89 | 209,63 |
| n90c3 | 90 | 180 | 3 | -993 | -993 | 30 | -1684 | -1462,84 | 287,6 | -1543,00 | -1265,98 | 184,80 | -1679 | -1435,15 | 338,6 |
| n90c4 | 90 | 180 | 4 | -912 | -912 | 13 | -1281,33 | -1186,64 | 196,73 | -1273,50 | -1083,58 | 130,30 | -1543,5 | -1229,89 | 257,47 |
| n100c3 | 100 | 200 | 3 | -1148 | -1148 | 20 | -1334,5 | -1272,82 | 282,47 | -1269,50 | -1172,33 | 141,67 | -1357 | -1282,29 | 223,47 |
| n100c4 | 100 | 200 | 4 | -1192,5 | -1192,5 | 31 | -1786,67 | -1557 | 317,03 | -1529,50 | -1330,16 | 143,93 | -1710,83 | -1518,38 | 354,2 |
| n200c3 | 200 | 400 | 3 | -2560 | -2560 | 750 | -2763 | -2647,16 | 456,8 | -2810,67 | -2652,54 | 391,73 | -2792,83 | -2619,87 | 310,73 |
| n200c4 | 200 | 400 | 4 | -2573 | -2573 | 466 | -3442,67 | -3288,14 | 587,17 | -3299,83 | -3122,59 | 351,40 | -3512,67 | -3245,08 | 359,73 |
| n300c3 | 300 | 600 | 3 | -3742 | -3742 | 2619 | -4250,5 | -4066,97 | 567,5 | -4183,00 | -3966,09 | 376,80 | -4252,33 | -4024,77 | 369,9 |
| n300c4 | 300 | 600 | 4 | -3952,5 | -3952,5 | 2027 | -4177,5 | -4031,54 | 632,03 | -4486,00 | -4054,05 | 426,57 | -4118 | -4003,42 | 382,9 |
| n400c3 | 400 | 800 | 3 | -4520 | -4520 | 8342 | -5006 | -4806,48 | 706,5 | -5105,33 | -4715,83 | 423,67 | -5026,17 | -4773,79 | 458,27 |
| n400c4 | 400 | 800 | 4 | -4756 | -4756 | 6107 | -5113,33 | -4877,78 | 736,63 | -4965,50 | -4820,02 | 505,23 | -4947,5 | -4822,32 | 488,27 |
| n500c3 | 500 | 1000 | 3 | -6972 | -6972 | 7364 | -7493,83 | -7268,29 | 896,4 | -7394,50 | -7160,23 | 555,00 | -7511,17 | -7188,6 | 543,83 |
| n500c4 | 500 | 1000 | 4 | 0 | 0 | 10002 | -6049 | -5848,18 | 826,97 | -6060,83 | -5764,18 | 560,97 | -5925 | -5758,51 | 538,17 |

**Table 7.** *p-valores* entre pares de algoritmos considerando a coluna *Méd.* da Tabela 6.

| | NVH | GVND1 | GVND2 | ACO |
|---|---|---|---|---|
| NVH | - | 0.0000 | 0.0003 | 0.0000 |
| GVND1 | - | - | 0.0020 | 0.6637 |
| GVND2 | - | - | - | 0.0061 |
| ACO | - | - | - | - |

**Table 8.** *p-valores* entre pares de algoritmos considerando a coluna *T(s)* da Tabela 6.

| | NVH | GVND1 | GVND2 | ACO |
|---|---|---|---|---|
| NVH | - | 0.0000 | 0.0015 | 0.0000 |
| GVND1 | - | - | 0.0015 | 0.0437 |
| GVND2 | - | - | - | 0.2059 |
| ACO | - | - | - | - |


[11] DAHLE, L., ANDERSSON, H., CHRISTIANSEN, M., AND SPERANZA, M. G. The pickup and delivery problem with time windows and occasional drivers. *Computers & Operations Research 109* (2019), 122–133.

[12] DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

[13] DORIGO, M., AND DI CARO, G. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (1999), vol. 2, IEEE, pp. 1470–1477.

[14] FAGNANT, D. J., AND KOCKELMAN, K. M. The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Trans-*

*portation Research Part C: Emerging Technologies 40* (2014), 1–13.

[15] FARHAN, J., AND CHEN, T. D. Impact of ridesharing on operational efficiency of shared autonomous electric vehicle fleet. *Transportation Research Part C: Emerging Technologies 93* (2018), 310–321.

[16] FEILLET, D., DEJAX, P., AND GENDREAU, M. Traveling salesman problems with profits. *Transportation science 39*, 2 (2005), 188–205.

[17] FEO, T. A., AND RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization 6*, 2 (1995), 109–133.

[18] FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association 32*, 200 (1937), 675–701.

[19] GAVISH, B., AND GRAVES, S. C. The travelling salesman problem and related problems.

[20] GENDREAU, M., GUERTIN, F., POTVIN, J.-Y., AND TAILLARD, E. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science 33*, 4 (1999), 381–390.

[21] GOLDBARG, M. C., AND LUNA, H. P. L. *Otimização combinatória e programação linear: modelos e algoritmos*. Elsevier, Rio de Janeiro, 2005.

[22] GU, Q.-P., LIANG, J. L., AND ZHANG, G. Algorithmic analysis for ridesharing of personal vehicles. *Theoretical Computer Science 749* (2018), 36–46.

[23] GUROBI, O. Gurobi optimizer reference manual version 6.5. *https://www.gurobi.com/documentation/6.5/refman* (2016).

[24] GUTIN, G., AND PUNNEN, A. P. *The traveling salesman problem and its variations*, vol. 12. Springer Science & Business Media, 2006.

[25] HANDOKO, S. D., CHUIN, L. H., GUPTA, A., SOON, O. Y., KIM, H. C., AND SIEW, T. P. Solving multivehicle profitable tour problem via knowledge adoption in evolutionary bi-level programming. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (2015), IEEE, pp. 2713–2720.

[26] HANSEN, P., AND MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European journal of operational research 130*, 3 (2001), 449–467.

[27] HELSGAUN, K. General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation 1*, 2-3 (2009), 119–163.

[28] HO, S. C., SZETO, W., KUO, Y.-H., LEUNG, J. M., PETERING, M., AND TOU, T. W. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological 111* (2018), 395–421.

[29] JABERI, N., AND RAFEH, R. On the linearization of zinc models. *Journal of Advances in Computer Research 5*, 4 (2014), 1–8.

[30] JACOBSON, S. H., AND KING, D. M. Fuel saving and ridesharing in the us: Motivations, limitations, and opportunities. *Transportation Research Part D: Transport and Environment 14*, 1 (2009), 14–21.

[31] JOZEFOWIEZ, N., GLOVER, F., AND LAGUNA, M. Multi-objective meta-heuristics for the traveling salesman problem with profits. *Journal of Mathematical Modelling and Algorithms 7*, 2 (2008), 177–195.

[32] LAHYANI, R., KHEMAKHEM, M., AND SEMET, F. Heuristics for rich profitable tour problems. In *2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)* (2013), IEEE, pp. 1–3.

[33] LI, B., KRUSHINSKY, D., REIJERS, H. A., AND VAN WOENSEL, T. The share-a-ride problem: People and parcels sharing taxis. *European Journal of Operational Research 238*, 1 (2014), 31–40.

[34] LI, Z., HONG, Y., AND ZHANG, Z. Do ride-sharing services affect traffic congestion? an empirical study of uber entry. *Soc. Sci. Res. Netw 2002* (2016), 1–29.

[35] LIN, C. A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources. *Computers & Operations Research 38*, 11 (2011), 1596–1609.

[36] LÓPEZ-IBÁÑEZ, M., DUBOIS-LACOSTE, J., CÁCERES, L. P., BIRATTARI, M., AND STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives 3* (2016), 43–58.

[37] MA, S., ZHENG, Y., AND WOLFSON, O. Real-time city-scale taxi ridesharing. *IEEE Transactions on Knowledge and Data Engineering 27*, 7 (2014), 1782–1795.

[38] PARRAGH, S. N., DOERNER, K. F., AND HARTL, R. F. A survey on pickup and delivery problems - part i: Transportation between customers and depot. *Journal für Betriebswirtschaft 58*, 1 (2008), 21–51.

[39] PARRAGH, S. N., DOERNER, K. F., AND HARTL, R. F. A survey on pickup and delivery problems - part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft 58*, 2 (2008), 81–117.

[40] PILLAC, V., GENDREAU, M., GUÉRET, C., AND MEDAGLIA, A. L. A review of dynamic vehicle routing problems. *European Journal of Operational Research 225*, 1 (2013), 1–11.

[41] PSARAFTIS, H. N. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science 14*, 2 (1980), 130–154.

[42] SIEGEL, S., AND CASTELLAN JR, N. J. *Nonparametric statistics for the behavioral sciences*, 2 ed. Mcgraw-Hill Book Company, New York, NY, 1988.

[43] SUN, P., VEELENTURF, L. P., DABIA, S., AND VAN WOENSEL, T. The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research 264*, 3 (2018), 1058–1073.

[44] YU, B., MA, Y., XUE, M., TANG, B., WANG, B., YAN, J., AND WEI, Y.-M. Environmental benefits from ridesharing: A case of beijing. *Applied energy 191* (2017), 141–152.

[45] ZHANG, M., WANG, J., AND LIU, H. The probabilistic profitable tour problem. *International Journal of Enterprise Information Systems (IJEIS) 13*, 3 (2017), 51–64.