UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HUGO VENÂNCIO CHAVES DE SOUZA

PROBLEMA DE COMPARTILHAMENTO DE VEÍCULOS COM RESTRIÇÕES FÍSICAS E SOCIAIS PERMITINDO DETERMINADOS TIPOS DE CICLOS: UMA ABORDAGEM HEURÍSTICA

RUSSAS

2018

HUGO VENÂNCIO CHAVES DE SOUZA

PROBLEMA DE COMPARTILHAMENTO DE VEÍCULOS COM RESTRIÇÕES FÍSICAS E SOCIAIS PERMITINDO DETERMINADOS TIPOS DE CICLOS: UMA ABORDAGEM HEURÍSTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientadora: Prof. Ms. Tatiane Fernandes Figueredo

RUSSAS

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S238p Souza, Hugo Venâncio Chaves de Souza.
Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais Permitindo Determinados Tipos de Ciclos: uma abordagem heurística / Hugo Venâncio Chaves de Souza Souza. – 2018.
40 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2018.
Orientação: Profa. Ma. Tatiane Fernandes Figueiredo.

1. Compartilhamento de Veículos. 2. Otimização Combinatória. 3. Heurística. I. Título.
CDD 005

HUGO VENÂNCIO CHAVES DE SOUZA

PROBLEMA DE COMPARTILHAMENTO DE VEÍCULOS COM RESTRIÇÕES FÍSICAS E SOCIAIS PERMITINDO DETERMINADOS TIPOS DE CICLOS: UMA ABORDAGEM HEURÍSTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Ms. Tatiane Fernandes Figueredo (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcio Costa Santos
Universidade Federal do Ceará (UFC)

Prof. Dr. Pablo Luiz Braga Soares
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Agradeço em primeiro lugar a minha orientadora Tatiane Fernandes Figueiredo, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos. A todos os professores que me acompanharam durante toda a graduação. Agradeço a minha mãe Vandeniza Chaves de Oliveira, heroína que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço. Ao meu pai Francisco Vanderlan de Souza que apesar de todas as dificuldades me fortaleceu e que para mim foi muito importante. Aos meus familiares em geral pela torcida e apoio. Ao grupo de pesquisa Núcleo de Estudo em aprendizado de Máquina e Otimização (NEMO) e por fim aos meus amigos que sempre estiveram comigo me apoiando e me dando forças para continuar.

“Faça o teu melhor, na condição que você tem, enquanto você não tem condições melhores, para fazer melhor ainda!”

(Mario Sergio Cortella)

RESUMO

Com o passar dos anos, tem-se notado um crescimento desenfreado no número de veículos pessoais em todo país, sendo este crescimento um dos grandes responsáveis pela geração de problemas ambientais e de mobilidade, principalmente nas grandes cidades. Uma solução para esse problema, muito utilizada por grande parte da população, é o compartilhamento de veículos entre pessoas com itinerários semelhantes em suas locomoções. Embora seja uma solução muito usual, o formato aplicado hoje pode gerar riscos à segurança dos envolvidos, visto que muitos compartilhamentos são realizados com indivíduos desconhecidos. Através da análise da localização física e social dos indivíduos envolvidos, este trabalho propõe um novo método para resolução deste problema. Para tal, é realizado um estudo sobre o Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais e apresentado um algoritmo heurístico para sua resolução. A técnica aplicada busca por soluções que procuram minimizar as distâncias entre os envolvidos, procurando também unir indivíduos com um alto nível de relação social. Dois grupos de teste foram criados para analisar o algoritmo apresentado. O primeiro experimento comparou os resultados obtidos com o método exato de Rodrigues (2018). Para as 39 instâncias testadas a heurística proposta apresenta, em média, um GAP de 37% utilizando a metade do tempo gasto pelo algoritmo exato.

Palavras-chave: Compartilhamento de veículos. Otimização combinatória. Heurísticas.

ABSTRACT

Over the years, there has been an unbridled growth in the number of personal vehicles throughout the country, and this is one of the main factors responsible for environmental and mobility problems, especially in big cities. One solution to this problem, used by a large part of the population, is the sharing of vehicles between people with similar itineraries. Although it is a very usual solution, the manner applied today can present risks to the security of those involved, since many shares are made with unknown individuals. Through the analysis of physical location and social of individuals, this work proposes a new solution for solving this problem. To this end, a study on the Problem of Vehicle Sharing with Physical and Social Restrictions is presented and methods for its through a heuristic approach. Applied techniques search for solutions that seek to minimize the distances between the involved ones, trying also to maximize the social relations between them. Two test groups were created to analyze the presented algorithm. The first experiment compared the results obtained with the exact method of Rodrigues (2018). For the 39 tested instances, the proposed heuristic presents, on average, a GAP of 37% using half the time spent by the exact algorithm.

Keywords: Ridesharing. Combinatorial optimization. Heuristics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

PCMVOV	Problema do Caminho Mínimo com Visita Obrigatória de Vértices
PCV-MCa	Problema do Caixeiro Viajante com Múltiplas Caronas
PCVFS	Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais

SUMÁRIO

1 INTRODUÇÃO

Dentre os problemas atuais enfrentados diariamente em grandes cidades, destaca-se a ampla quantidade de veículos em trânsito, sendo responsáveis por diversos problemas ambientais e de locomoção (COSTA, 2003). Entre várias possíveis soluções, com objetivo de diminuir esses impactos, destaca-se a de que pessoas, com itinerários semelhantes nas suas locomoções, conduzindo veículos com vagas ociosas poderiam realizar compartilhamentos, procurando assim, reduzir os efeitos deste problema. Contudo, essa solução necessita de cuidados com a segurança, visto que os indivíduos envolvidos podem ser desconhecidos.

Aplicativos tem sido criados na tentativa de facilitar o compartilhamento dessas vagas, como o BlaBlaCar () e o Carona Fácil (). Porém, os mesmos não levam em consideração as distâncias físicas entre os participantes e não realizam qualquer análise sobre perfil social dos envolvidos. Considerando trabalhos relacionados a este problema, obtidos por comunidades científicas, alguns problemas clássicos da literatura podem ser utilizados para solucionar o Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais (PCVFS), como, por exemplo, o Problema do Caminho Mínimo com Visita Obrigatória de Vértices (PCMVOV)(ANDRADE, 2016) que, assim como o trabalho proposto, tem por objetivo encontrar o menor caminho em um grafo visitando obrigatoriamente determinados vértices, que podem ser considerados como participantes do compartilhamento, porém, não leva em consideração as relações sociais entre os indivíduos participantes, não levando em consideração a segurança do compartilhamento.

Além do compartilhamento de vagas ociosas em veículos, este problema também objetiva minimizar a distância percorrida do trajeto, ao mesmo tempo que procura também minimizar relacionamentos de não afinidade, ou seja, dado um par de indivíduos, uma função representará a afinidade que estes indivíduos possuem. Quanto menor o valor dado pelo função, maior a afinidade entre os mesmos. Para tal, neste trabalho, é apresentado um algoritmo heurístico fundamentado no algoritmo de Busca Dispersa, criado por Glover (1977).

1.1 Objetivos

1.1.1 Objetivos Gerais

Obter soluções para o Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais através de técnicas heurísticas.

1.1.2 Objetivos Específicos

- Criar um algoritmo heurístico para o Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais;
- Efetuar uma análise do desempenho da heurística apresentada.

1.2 Metodologia

Dada a natureza aplicada da pesquisa realizada, o trabalho será desenvolvido com base na elaboração de um estudo de pesquisa operacional padronizado da seguinte forma:

- Estudo de ferramentas para resolução de problemas de otimização combinatória com foco em algoritmos heurísticos;
- Definição das restrições e objetivos esperados para resolução problema estudado;
- Desenvolvimento de um procedimento computacional a fim de derivar soluções para o problema baseado em algoritmos heurísticos;
- Em seguida, o algoritmo proposto será alimentado com as instâncias existentes na literatura, a fim de comparar os seus desempenhos. O estudo será explicativo e seguirá uma abordagem quantitativa, utilizando-se do método de simulação. Neste método, o algoritmo heurístico será utilizado para resolução do Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais.

1.3 Organização do Trabalho

O restante deste trabalho se divide em 7 capítulos. No Capítulo 2 é apresentado uma contextualização do Problema de Compartilhamento de Veículos na sociedade atual. No Capítulo 3 são apresentados os conceitos bases necessários para resolução do problema proposto. O Capítulo 4 apresenta os trabalhos relacionados ao tema da monografia. O Capítulo 5 define matematicamente o Problema do Compartilhamento de Veículos com Restrições Físicas e Sociais. No Capítulo 6 apresenta os resultados obtidos no trabalho. Por fim, o Capítulo 7 apresenta as conclusões e os trabalhos futuros.

2 PROBLEMA DE COMPARTILHAMENTO DE VEÍCULOS

O problema de compartilhamento de veículos é definido com base na ideia de que pessoas que realizam itinerários semelhantes nas suas locomoções diárias, conduzindo os veículos individualmente, poderiam realizar um compartilhamento. Essa solução requer certos cuidados, pois o compartilhamento com indivíduos desconhecidos pode gerar riscos à segurança dos envolvidos.

Uma das principais motivações para o estudo de tal problema é a existência de um grande número de veículos particulares com vagas ociosas nos engarrafamentos dos grandes centros urbanos. O compartilhamento de veículos tem potencial para reduzir problemas de tráfego, custos, impactos ambientais, e otimizar a capacidade já existente de veículos em circulação (ARAÚJO, 2016). Contudo, apesar de todas as vantagens do compartilhamento de veículos, não se utiliza muito dele, principalmente pela da falta de segurança, visto que nem sempre há muitas informações pessoais sobre os indivíduos participantes do compartilhamento. Uma das propostas, muito utilizadas nos dias atuais, para resolução da falta de segurança é a aplicação de mineração de dados em redes sociais.

Uma rede social é composta por pessoas ou organizações, conectadas por um ou vários tipos de relações. Através da aplicação de técnicas de mineração de dados em redes sociais é possível classificar as relações entre os indivíduos participantes da rede com o intuito de utilizar essas classificações para sanar possíveis problemas de segurança, por exemplo, relacionados ao problema de compartilhamento de veículos. Atualmente, alguns aplicativos tem utilizado redes sociais com a finalidade de garantir a segurança dos usuários, como por exemplo, o BlaBlaCar () criado na França em 2006 ou Carona Fácil () que conta com um grupo de mais de 50 mil usuários (NETO; ANDRADE, 2016). Porém, em ambos os casos, existe apenas a informação se os indivíduos envolvidos são, ou não, amigos na plataforma *Facebook*, sem levar em consideração as distâncias físicas dos mesmos. Até o presente momento, para o nosso maior conhecimento, nenhum aplicativo do mercado utiliza mineração de dados juntamente com métricas de distância.

3 REFERENCIAL TEÓRICO

Para o desenvolvimento da heurística abordada neste trabalho são necessários alguns conhecimentos iniciais na área de Teoria dos Grafos, de Heurísticas e Meta-heurísticas, que serão apresentados nas subseções 3.1, 3.2 e 3.3, respectivamente.

3.1 Teoria dos Grafos

Um *grafo* $G = (V(G), E(G))$ consiste de um conjunto finito não vazio $V(G)$ de elementos, chamados *vértices*, e um conjunto de pares não-ordenados de elementos distintos de $V(G)$, chamado *arestas*, isto é, $E(G) \subseteq \{\{u,v\} | u,v \in V(G), u \neq v\}$. Algumas vezes, quando o grafo estiver claro pelo contexto, utilizamos V e E para $V(G)$ e $E(G)$ respectivamente, para simplificar a notação. Se $e = \{u,v\}$ é uma aresta, dizemos que e incide em u e em v; que *u e v são seus extremos*; e que *u e v são adjacentes*. Para simplificar a notação, denotamos uma aresta $e = \{u,v\}$ por uv.

A *vizinhança* ou *adjacência* de um vértice v em um grafo G é o conjunto contendo todos os vértices adjacentes a v. A vizinhança é frequentemente denotada $N_G(v)$ ou simplesmente $N(v)$ quando o grafo está claro pelo contexto. A vizinhança de um subconjunto de vértices $W \subseteq V$ é $N(W) = \bigcup_{v \in W} N(v)$. Se V' é um subconjunto de vértices em $V(G)$, denotamos por $\delta(V')$ o conjunto de arestas em $E(G)$ com um extremo em V' e o outro em $V \setminus V'$. Se G e G' são dois grafos tais que $V(G') \subseteq V(G)$ e $E(G') \subseteq E(G)$, então G' é dito um *subgrafo* de G e escrevemos $G' \subseteq G$.

Um *grafo orientado (digrafo)* $D = (V,A)$ consiste de um conjunto finito não-vazio $V(D)$ de elementos, chamados *vértices* ou *nós*, e um conjunto $A(D)$ de elementos, chamados *arcos* ou *arestas orientadas*, que, em digrafos sem loops, são pares ordenados de vértices distintos, isto é, $A(D) \subseteq \{(u,v) | u,v \in V(D), u \neq v\}$. Se $a = (u,v)$ é um arco, dizemos que a incide em u e em v; que u e v são seus extremos. Também nos referimos a u como *início* ou *origem* e a v como *término* ou *destino* do arco a.

Todos os conceitos definidos para grafos que não envolvem a noção de orientação se aplicam analogamente para os grafos orientados. Embora usemos preferencialmente a notação $D = (V,A)$ para um grafo orientado, vamos usar também $G = (V,E)$, quando não houver prejuízo para o entendimento, e neste caso E será um conjunto de arcos. Em um *grafo* orientado $D = (V,A)$, um conjunto não-vazio de arcos $P = \{\alpha_1, \alpha_2, ..., \alpha_k\}$, onde $\alpha_i = (v_i, v_j) \in A$, e

$v_i \neq v_j$ para $i \neq j$, é chamado caminho. Dizemos que P é um (v_1, v_k)-caminho; e que tem comprimento $k-1$. Se $(v_k, v_1) \in A$ então $C := P \cup \{(v_k, v_1)\}$ é chamado circuito. Removendo-se a orientação (dos arcos) temos analogamente o conceito de caminho e ciclo em um grafo. A distância entre dois vértices u e v em um grafo (não orientado) é o comprimento de um (u, v)-caminho de menor comprimento possível; se não existe um tal caminho, então dizemos que a distância é infinita.

3.2 Problema do Caminho Mínimo

O Problema do Caminho Mínimo (*Shortest Path Problem*) tem por objetivo encontrar o menor caminho entre dois vértices. O menor caminho pode ser definido de diversas formas de acordo com a aplicação do problema, como por exemplo, menor custo entre dois pontos, ou a menor distância, ou ainda o menor tempo de viagem, entre outros (MÉNDEZ; GUARDIA, 2008). O Problema do Caminho Mínimo tem múltiplas aplicações e ainda múltiplas abordagens para resolução. Neste trabalho, utilizamos o algoritmo de Dijkstra que resolve o problema do caminho mínimo em grafos orientados ou não orientados, com arestas de peso não negativas, em tempo polinomial. O seu funcionamento ocorre como descrito no Algoritmo 1. Segue abaixo uma descrição do algoritmo.

Dado um grafo orientado $D(V,A)$, onde cada arco $(i, j) \in A$ está associado a um custo não-negativo c_{ij} e um vetor L que armazena a distância entre os vértices de $V(D)$ e o vértice inicial a, o algoritmo inicializa todas as posições, em L, com valores infinitos, para cada vértice em V que não esteja no conjunto S, é verificada a distância entre ele e o vértice selecionado, se for menor, o valor em L é atualizado, caso contrário o algoritmo segue para o próximo vértice. Por fim, um conjunto S conterá os vértices cujo caminho mínimo para a é conhecido em cada iteração. Ao final, como resultado, o algoritmo retornará todas as menores distâncias para os vértices pertencentes a $V - \{a\}$.

S contém os vértices cujo caminho mínimo para a é conhecido em cada iteração.

Algoritmo 1: ALGORITMO DE DIJKSTRA

Entrada: $G(V,E)$, c_{ij}, L , $S \subseteq V$, $a \in V$

Saída: L - atualizado com as menores distâncias possíveis em relação ao vértice a

início

S ← a

L[a] ← 0

escolhido ← a

repita

para *cada* $v \in \{V - S\}$ **faça**

$custo \leftarrow L[escolhido] + c_{escolhido,v}$

se $custo < L[v]$ **então**

$L[v] \leftarrow custo$

fim

fim

escolhido ← $min_{u \in \{V-S\}}(L[u])$

S ← escolhido

até *que S tenha todos os vértices de V*;

fim

retorna L

3.3 Heurísticas e Meta-heurísticas

Algoritmos heurísticos, ou heurísticas, são métodos para obtenção de soluções aproximadas, que podem conter o valor ótimo ou valores próximos do mesmo, em problemas de otimização. O desenvolvimento de heurísticas surge em resposta à impossibilidade de se resolver, satisfatoriamente, diversos problemas de otimização NP-Difíceis (ARROYO *et al.*, 2002). Já uma meta-heurística é uma estratégia de busca, não específica para um determinado problema, que tenta explorar eficientemente o espaço das soluções viáveis desse problema. São algoritmos aproximados que incorporam mecanismos para evitar confinamento em mínimos ou máximos locais.

3.3.1 Busca Dispersa

A busca dispersa, ou *Scatter Search* em inglês, é uma meta-heurística evolutiva (baseada em populações) muito efetiva em diversos problemas de otimização discreta, sendo que cada um de seus elementos podem ser implementados com uma variedade de formas e graus de liberdade. Nesta meta-heurística opera-se em um conjunto de soluções, normalmente chamado conjunto de referência ou *Refset*, combinando as soluções existentes para geração de novas possíveis soluções. O objetivo principal é que as soluções geradas em cada iteração sejam melhores que as geradas em iterações anteriores (LIMA, 2012).

Partindo da crença de que informações melhores podem ser obtidas se tratadas de forma integrada, do que quando tratadas isoladamente. O *Refset* guarda "boas" soluções encontradas durante o processo de busca, uma solução é considerada "boa" não somente levando em consideração seus critérios de qualidade, mas também sua diversidade em comparação com as outras (SOSA *et al.*, 2007). Os critérios de qualidade são definidos de acordo com o problema no qual a meta-heurística é aplicada. É demonstrada a seguir uma descrição mais detalhada do algoritmo e logo em seguida um pseudocódigo do busca dispersa.

A Busca Dispersa é composta basicamente por 5 etapas:

1. No Algoritmo 2 da linha 1 até a linha 7 está representada a etapa de geração de soluções iniciais: nesta etapa é gerado um conjunto P de soluções de tamanho $Psize$.
2. Na linha 8 do algoritmo está representada a etapa de melhoria: é aplicada uma função para melhorar as $Psize$ soluções inicialmente encontradas. Caso a solução que está sendo avaliada seja inviável, o método tentará transformá-la em uma solução viável. Se a solução for viável, o método tentará melhorá-la.
3. Nas linhas 9, 10, 20 e 21 ocorre a geração do conjunto de referência: nesta etapa o conjunto *Refset* é construído ou atualizado.
    a) Construção (linhas 9 e 10) – A partir do conjunto P se extrai *Refset*, combinando $b1$ soluções de alta qualidade com as $b2$ soluções que contenham alta diversidade, durante o processo de busca os componentes de *Refset* são usados para gerar novas soluções mediante a combinação de elementos contidos nesse conjunto.
    b) Atualização (linhas 20 e 21) – A atualização de *Refset* acontece quando novas soluções que atendem os requisitos para ingressar em *Refset* são encontradas, onde novamente são selecionadas $b1$ das melhores soluções e $b2$ das mais diversificadas, assim, *Refset* mantém seu tamanho constante, visto que as soluções ingressantes subs-

tituem outras já existentes, enquanto as soluções pertencentes ao mesmo melhoram durante o processo de busca. Para atualizar o Refset tem-se que considerar o momento em que a atualização deve ser realizada (estática ou dinâmica). A atualização se diz estática quando a mesma é realizada após testadas todas as possíveis combinações no Refset; se diz dinâmica quando imediatamente após alguma combinação a solução resultante é inserida no Refset, sempre que atenda os requisitos necessários para ingressar no conjunto.

4. Na linha 12 ocorre a geração de subconjuntos: utilizando o *Refset* para criar diferentes subconjuntos de soluções que serão utilizadas na etapa de combinação
5. Da linha 13 até a linha 17 é realizada a combinação de soluções: a partir dos subconjuntos gerados na Etapa 4, é efetuado uma combinação entre subconjuntos com o objetivo de gerar novas soluções. A etapa de combinação é definida de acordo com o problema no qual a meta-heurística é aplicada, uma vez que está diretamente relacionada com a representação da solução.

Algoritmo 2: Descrição geral do Busca Dispersa

Entrada: $Psize, G, v, s, t, w, f, \alpha, \beta, b1, b2, n, m$
Dados: *P*, *RefSet*, *pool*

```
1  P ← ∅
2  repita
3      solucao ← gerarSolucao()
4      se solucao ∉ P então
5          P ← solucao
6      fim
7  até tamanho de P seja igual a Psize;
8  melhorar(P)
9  RefSet ← melhores(b1, P)
10 RefSet ← diversificadas(b2, P)
11 repita
12     gerarSubConjuntos(RefSet)
13     para cada subconjunto r de RefSet faça
14         para cada par i, j de soluções em r faça
15             pool ← combinacao(i, j)
16         fim
17     fim
18     melhorar(pool)
19     RefSet ← pool
20     RefSet ← melhores(b1, RefSet)
21     RefSet ← diversificadas(b2, RefSet)
22 até condicao de parada;
23 retorna melhor(RefSet)
```

4 TRABALHOS RELACIONADOS

Este capítulo descreve alguns trabalhos da literatura mais relevantes para a contextualização do problema proposto nesta monografia. Nas Seções 4.1 e 4.2 são apresentados problemas semelhantes encontrado na literatura.

4.1 Problema do Caixeiro Viajante Com Múltiplas Caronas

O Problema do Caixeiro Viajante com Múltiplas Caronas (PCV-MCa) é um problema que resulta da união dos problemas do Caixeiro Viajante com problema de compartilhamento de veículos (ARAÚJO, 2016). Considerando os elementos presentes no Caixeiro Viajante acrescenta-se a necessidade de diminuir o custo da viagem que agora será restringida pela obrigatoriedade de passagem pelas localidades dos indivíduos que estarão compartilhando o veículo. Note que o trajeto do motorista será adaptado em função dos trajetos dos passageiros, alcançando o custo mínimo através do partilhamento de custos entre todos os envolvidos. Assim como neste trabalho, o foco do problema é o compartilhamento de veículos com redução de custos, porém não há considerações referentes a segurança do compartilhamento, como por exemplo, utilização de informações sobre as pessoas que receberão a carona e como as mesmas serão encontradas pelo Caixeiro Viajante.

4.2 Problema do Caminho Mínimo com Visita Obrigatória de Vértices

Dado um grafo $G = (V, A)$, uma matriz de custo C_{uv} com o custo entre os vértices u e v, dois vértices s e t, ambos pertencentes a V e um conjunto P que está contido em $V - \{s, t\}$, o problema do caminho mínimo com visita obrigatório (PMC-VOV) consiste em encontrar o menor caminho entre s e t que visite, somente uma vez, cada vértice de P (ANDRADE, 2016). Nosso trabalho apresenta uma generalização para este problema, visto que, para instâncias onde a afinidade de todos os indivíduos são de mesmo grau, o número obrigatório de vértices a ser visitado pode ser visto como as localizações dos indivíduos que deverão participar do compartilhamento.

4.3 Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais

Dado um grafo $G = (V,A)$, um vértice inicial s e um vértice final t, um grupo de vértices $V' \in V(G)$ com possíveis participantes do compartilhamento, o PCVFS consiste em encontrar o menor caminho entre s e t que visite uma quantidade w de vértices de V', procurando minimizar o caminho e minimizar relacionamentos de não afinidade, porém a abordagem exata utilizada não permite ciclos na composição do caminho, fazendo com que soluções viáveis com ciclo sejam consideradas inviáveis. Nosso trabalho apresenta uma forma de aceitar esses tipos de ciclo, com os mesmos objetivos.

5 PROBLEMA DE COMPARTILHAMENTO DE VEÍCULOS COM RESTRIÇÕES FÍSICAS E SOCIAIS - CONSIDERANDO SOLUÇÕES QUE POSSUAM CICLOS

Ao contrário do PCV-MCa, o PCVFS propõe uma solução que considere a afinidade entre os envolvidos de um compartilhamento com o intuito de garantir alguma forma de segurança. Além de procurar minimizar a distância percorrida do trajeto, o PCVFS procura também minimizar relacionamentos de não afinidade, ou seja, dado um par de indivíduos, uma função representará a afinidade que estes indivíduos possuem. Quanto menor o valor dado pelo função, maior a afinidade entre os mesmos. Para descrevê-lo utiliza-se um digrafo que representa o mapa utilizado para locomoção dos indivíduos participantes do compartilhamento. Mais formalmente:

Problema 5.0.1 *O Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais (PCVFS)*

Entrada: Uma tupla $(D, V', s, t, w, f, d, \alpha, \beta)$*, onde:*

- $D = (V, A)$ *é um digrafo conexo, onde* $V(D)$ *são as localizações do mapa e* $A(D)$ *são arestas rotuladas com a distância entre u e v;*
- V' *é um subconjunto de* $V(D)$*, tal que* $V' \subset V(D)$ *e indica os indivíduos* v' *que podem receber carona;*
- s *é um vértice, tal que* $s \in V(D)$*, representando a localização inicial de quem dá carona;*
- t *é um vértice, tal que* $t \in V(D)$*, representando o destino dos indivíduos;*
- w *é uma constante, tal que* $w \in \mathbb{N}$ *representa o número de pessoas que receberão carona;*
- *Função* $f : V' \to \mathbb{Z}_+$ *associadas a* V'*, onde quanto menor o valor de* $f(u)$*, maior é o grau de afinidade entre os vértices s e u;*
- *Função* $d : A \to \mathbb{R}_+$ *associadas a* $A(D)$*, que denota o peso do arco* $uv \in A(D)$ *representando a distância entre os vértices u e v.*
- α *e* β *são multiplicadores da função objetivo, tal que* $\alpha + \beta = 1$ *e definem a prioridade de cada medida.*

Questão*: Determinar* $s-t$ *passeio, onde:*

1. *Exatamente w vértices pertencentes a* V' *estejam no passeio* $s-t$*;*
2. *O somatório dos valores* $f(v) \in V'$ *e* $d(uv) \in A$*, para todos os vértices v e arestas uv pertencentes ao passeio* $s-t$ *sejam minimizados.*

É importante mencionar, que os algoritmos exatos criados para o problema apresen-

tado na seção 4.3 possuem restrições que não permitem ciclos, por serem baseados no problema de caminho mínimo com visita obrigatória de vértices apresentados na seção 4.2. Esta restrição faz com que soluções, como as apresentadas nas Figuras 1 e 2, que contenham ciclos não sejam consideradas soluções viáveis. As figuras a seguir apresentam exemplos, onde o único vértice a receber carona é o vértice v_2, ou seja $V' = \{v_2\}$. A Figura 1 apresenta um exemplo de ciclo em que há uma repetição no vértice $v1$ e na Figura 2 um exemplo de ciclo em que ocorre uma repetição da aresta (s, v_1). Ambos exemplos poderiam ser consideradas como soluções viáveis para o problema real de compartilhamento de veículos, porém não são aceitas pelo algoritmo exato presente na literatura.

Figura 1 – Grafo com ciclo entre v_1 e v_2 sendo necessária a repetição de um vértice

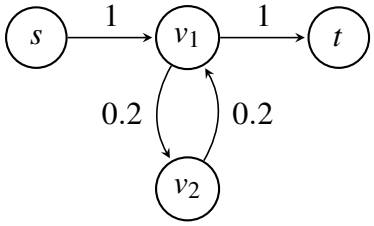
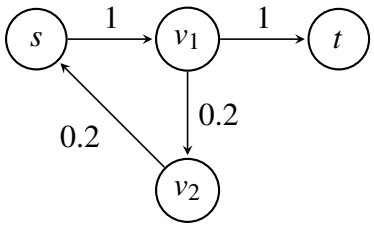


Figura 2 – Grafo com ciclo entre s, v_1 e v_2 sendo necessária a repetição de arestas



A solução viável da Figura 1 seria: partindo do vértice s para o vértice v_1, passando para o vértice v_2, contabilizando o indivíduo localizado neste vértice para participar do compartilhamento, retornando para o vértice v_1 e, por fim, finalizando o passeio em t. A solução viável da Figura 2 seria: partindo do vértice s para o vértice v_1, passando para o vértice v_2, contabilizando o indivíduo localizado neste vértice para participar do compartilhamento, retornando ao vértice s, logo após, passando novamente pelo vértice v_1 e, por fim finalizando o passeio em t.

5.1 Uma Heurística Fundamentada em Busca Dispersa para o PCVFS

Descreveremos nas próximas seções um algoritmo heurístico fundamentado no algoritmo de Busca Dispersa. Para tal, considera-se os seguintes parâmetros e definições:

- $m = 5$ (quantidade de gerações);

- $n = 1$ (quantidade de indivíduos por geração);
- $b1 = 0,5$ (quantidade de soluções de alta qualidade);
- $b2 = 0,5$ (quantidade de soluções de baixa qualidade);
- $Psize = 10$ (tamanho da população inicial);
- $RefSet = 5$ (tamanho do conjunto de referência);
- $\gamma = 3$ (número de máxima de gerações sem alteração na qualidade da solução - convergência);

5.1.1 Critérios de qualidade de uma solução

Para obtermos o valor do critério de qualidade da solução fazemos o seguinte cálculo $\alpha * C + \beta * F$, onde $\alpha + \beta = 1$, onde:

- O valor da distância C é obtido a partir do vetor P com os vértices visitados e da matriz de adjacência do grafo G, é calculado o custo de percorrer todos os vértices de P e o valor total é atribuído a C;
- O valor de afinidade F é obtido da seguinte forma, para cada carona verificamos seu valor em f, todos os valores de afinidade obtidos são somados, o que nos deixa com o valor da afinidade entre os participantes, valor este que é atribuído a F;
- Os valores de α e β, que são fornecidos pelas instâncias.

5.2 Representação de uma Solução

Neste trabalho, uma solução é representada por dois vetores. O vetor P apresenta a ordem dos vértices que participam do passeio, enquanto o vetor binário R indica quais vértices do passeio compartilharão um veículo. Os valores de custo do caminho C, afinidade das caronas F, assim como o valor total da qualidade de uma solução Q são calculados a partir de P e R. A Figura 3 representa uma solução viável para o grafo apresentado na Figura 1.

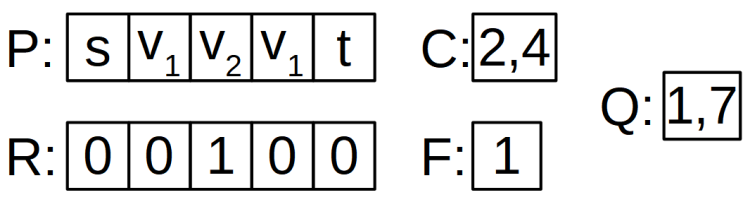


Figura 3 – Representação de uma solução de acordo com o grafo apresentado na Figura 1

5.3 Heurística Baseada em Busca Dispersa

Considerando como entrada uma tupla $(D, V', s, t, w, f, d, \beta, \alpha)$, ou seja, uma entrada do PCVFS e os parâmetros de configuração do algoritmo de Busca Dispersa ($b1, b2, n, m, Psize$, $RefSet, \gamma$) definidos na seção 5.1, é apresentado a seguir, o pseudoalgoritmo geral da heurística proposta neste trabalho (Algoritmo 3), assim como uma descrição detalhada de cada uma das suas etapas.

Algoritmo 3: Descrição geral do Busca Dispersa para *PCVFS*

Entrada: $Psize, G, v, s, t, w, f, \alpha, \beta, \gamma, b1, b2, n, m$

```
1  P ← criarPopulacaoInicial(Psize, G, v, s, t, w, f, α, β)
2  melhor ← ∞
3  RefSet ← melhores(P, b1) + piores(P, b2)
4  se w > 1 então
5      repita
6          pool ← ∅
7          repita
8              novoIndividuo ← crossover(G, w, RefSet, f, α, β)
9              pool ← pool + novoIndividuo
10         até n vezes;
11         RefSet ← Refset + pool
12         RefSet ← melhores(P, b1) + piores(P, b2)
13         se melhor > RefSet[0] então
14             melhor ← RefSet[0]
15             convergencia ← 0
16         senão
17             convergencia ← convergencia + 1
18             se convergencia = γ então
19                 retorna melhor
20             fim
21         fim
22     até m vezes;
23 fim
24 retorna melhor
```

5.3.1 Geração da População

Na linha 1 do algoritmo encontra-se a função *criarPopulacaoInicial(*$Psize, G, v, s, t, w, f, \alpha, \beta$*)*, onde são gerados *Psize* indivíduos para que os processos de combinação sejam possíveis. Para tal, utiliza-se uma aplicação controlada do algoritmo de Dijkstra. Cada indivíduo sairá do vértice s com objetivo de chegar no vértice t e aceitará w compartilhamentos aleatoriamente selecionadas, sem repetição. Estes compartilhamentos serão selecionadas escolhendo vértices pertencentes a V'. Ao ser selecionado um vértice $v_j \in V'$ para receber carona, o mesmo é removido da lista de opções de seleção, para evitar que dois compartilhamentos sejam contabilizados para um mesmo vértice. Como o algoritmo de Djikstra não aceita ciclos, ele será aplicado no grafo em partes. A Figura 5 apresenta um exemplo de aplicação do algoritmo.

Partindo do vértice s definiremos nossa primeira carona, que será selecionada aleatoriamente no conjunto V', que, no exemplo, contém os vértices 1 e 3, ao ser escolhido um vértice aplicamos então o algoritmo de Dijkstra para encontrar o caminho de s até o vértice selecionado, no exemplo o vértice 1, em seguida é selecionado aleatoriamente outro vértice de V' e novamente é aplicado Dijkstra entre a ultima carona selecionada e a nova carona selecionada, repetindo até que w caronas tenham sido selecionadas. Quando se tem todas as w caronas, é executado o algoritmo de Djkstra da última carona até o vértice t, obtendo assim um passeio que permite ciclos em caronas, mas não perite ciclos em outras ocasiões.

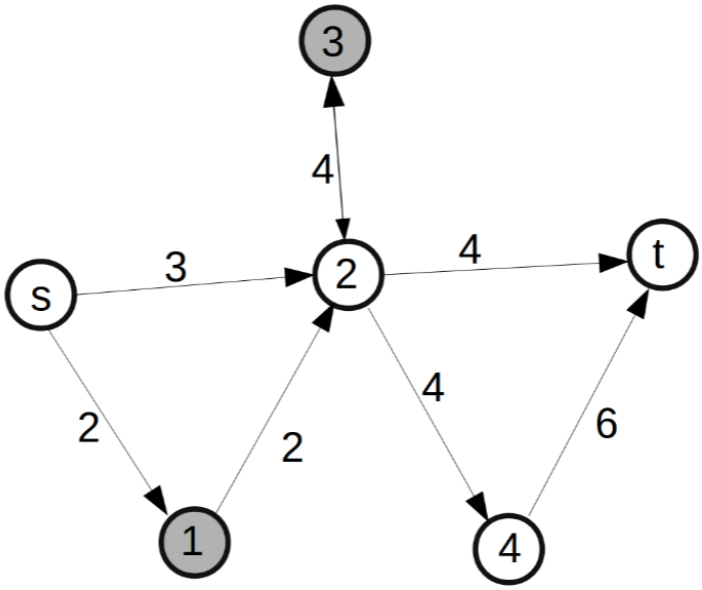


Figura 4 – Exemplo de grafo com caronas que podem gerar ciclos

Durante o percurso, os vértices escolhidos para fazer parte do passeio serão ar-

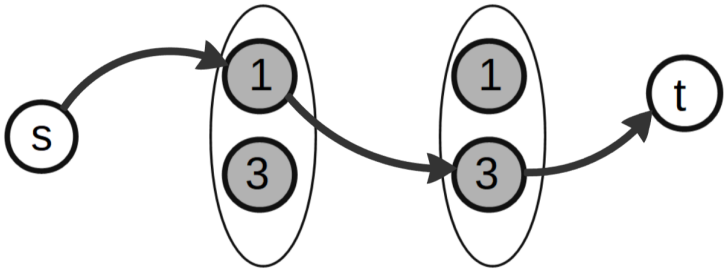


Figura 5 – Representação da aplicação do algoritmo de Dijkstra para obtenção de uma solução inicial no grafo da Figura 4

mazenados na estrutura P definida na seção 5.2, gerando assim uma solução viável para o problema.

5.3.2 Atualização do Conjunto de Referência

Na linha 3 encontram-se as funções *melhores(*$P, b1$*)* e *piores(*$P, b2$*)*, responsáveis pela geração do conjunto de referência. Para tal, é realizada uma ordenação, considerando os valores de qualidade das soluções do conjunto P, sendo escolhidos $b1 * RefSet$ melhores soluções e $b2 * RefSet$ piores soluções. Note que nas linhas 11 e 12 são efetuadas as mesmas atualizações definidas acima.

5.3.3 Geração de Novos Indivíduos

Na linha 8 realiza-se a criação de um novo indivíduo através de uma operação de *Crossover*. A seguir, na linha 9 atualiza-se o *pool*, conjunto de auxiliar que armazena temporariamente os novos indivíduos gerados.

5.3.3.1 Crossover

A partir do conjunto *RefSet*, são sorteados dois indivíduos para geração de um novo indivíduo a partir de combinações dos mesmos. Esta combinação é feita da seguinte forma, é definido uma partição utilizada em ambos indivíduos, neste algoritmo, considera que cerca de $\lfloor w/2 \rfloor$ vértices pertencentes a V' devem fazer parte de cada uma das 2 partições. Após separadas as partições, são verificadas as caronas de ambas, afim de verificar se uma carona qualquer de uma parte se repete na outra. Após isso, para a junção das duas partes, é realizada a verificação se existe um caminho direto entre a última carona do primeiro indivíduo e a primeira carona do segundo indivíduo, caso não haja, aplica-se o algoritmo de Dijkstra entre a ultima carona do

primeiro indivíduo e a primeira carona do segundo indivíduo, para corrigir os custos da nova solução gerada. Desta forma, gera-se um novo indivíduo, com uma possível qualidade de solução diferente dos indivíduos geradores.

5.3.4 Critério de Parada do Algoritmo

O algoritmo finaliza quando houver ocorrência de um dos seguintes casos:

- quando o número de gerações for igual a m; ou
- quando atingir o limite γ (convergência);

6 RESULTADOS

Neste capítulo é apresentado os resultados da implementação baseada no Busca Dispersa, descrito na seção 5.3, assim como uma avaliação comparativa com os algoritmos da literatura realizada através de experimentos computacionais e a análise de seus resultados. Na seção 6.1 é apresentado o ambiente computacional utilizado para realização dos testes. Na seção 6.2 uma breve definição das instâncias utilizadas na realização dos testes computacionais, seguida pelos resultados obtidos da heurística implementada e, por fim, uma comparação com os resultados presentes na literatura.

6.1 Configuração do Ambiente Computacional

A implementação do algoritmo heurístico, presente neste trabalho, faz uso do SageMath 8.2. Neste software matemático está presente a biblioteca denominada Graph que permite o desenvolvimento utilizando a linguagem de programação Python . Os experimentos realizados foram executados utilizando uma máquina com processador de quatro núcleos de 2.8 GHz, 8.0GB de memória RAM e sistema operacional Windows de 64 bits.

6.2 Instâncias

Para realização dos testes computacionais com a heurística, proposta neste trabalho, foram utilizadas instâncias criadas por Rodrigues (2018). Das 60 instâncias apresentadas foram utilizadas apenas 39 instâncias para os experimentos computacionais deste trabalho. A justificativa para o uso de um número reduzido de instâncias se dá pelo fato que o tempo para resolução das instâncias não escolhidas. Estas não ultrapassam o valor de 2 segundos, ou seja, podendo ser consideradas instâncias de fácil resolução, não havendo assim necessidade de aplicação de técnicas heurísticas para resolução deste tipo de instância.

Sobre os padrões das instâncias, estas se diferem apenas sobre o número de vértices, onde $|V| \in \{100, 120, 140, 160, 180, 200\}$. Para todas as instâncias testadas, o número de caronas que devem ser fornecidas é definido por $w \in \{1, 2, 3, 4\}$, os valores da função de afinidade de cada vértice $v \in V'$ é definido por $f \in \{1, 2, ..., 10\}$ e os valores da função de distância eram definidos por $d \in \{1, 2, ..., 6\}$.

6.3 Estudo Comparativo Entre Modelo Exato e Heurístico

A priori foram realizados testes comparando os resultado obtidos com o modelo exato Rodrigues (2018) e a heurística proposta neste trabalho. As Figuras 6 a 9 apresentam os dados dos resultados computacionais considerando as configurações de parâmetros apresentado na seção 5.1.

Na Tabela 1 a seguir estão presentes: a coluna do parâmetro analisado (*param*), a coluna com o valor mínimo do parâmetro (*min*), a coluna com a média do valor do parâmetro (*med*) e a coluna com o valor máximo do parâmetro (*max*), nas linhas temos a primeira com o tempo necessário para encontrar o valor ótimo (tp_{OPT}), a segunda com o tempo que a heurística passou executando (tp_{BD}) e a coluna com o *GAP*, onde:

$$GAP = (sol_{BD} - OPT)/OPT$$

Tabela 1 – Tabela com os valores mínimos, médios e máximos obtidos após a execução das instâncias com ambos algoritmos

Parâmetro	min	med	max
tp_{OPT}	2,14	5,47	8,62
tp_{BD}	1,06	2,52	5,31
GAP	0,03	0,37	0,70

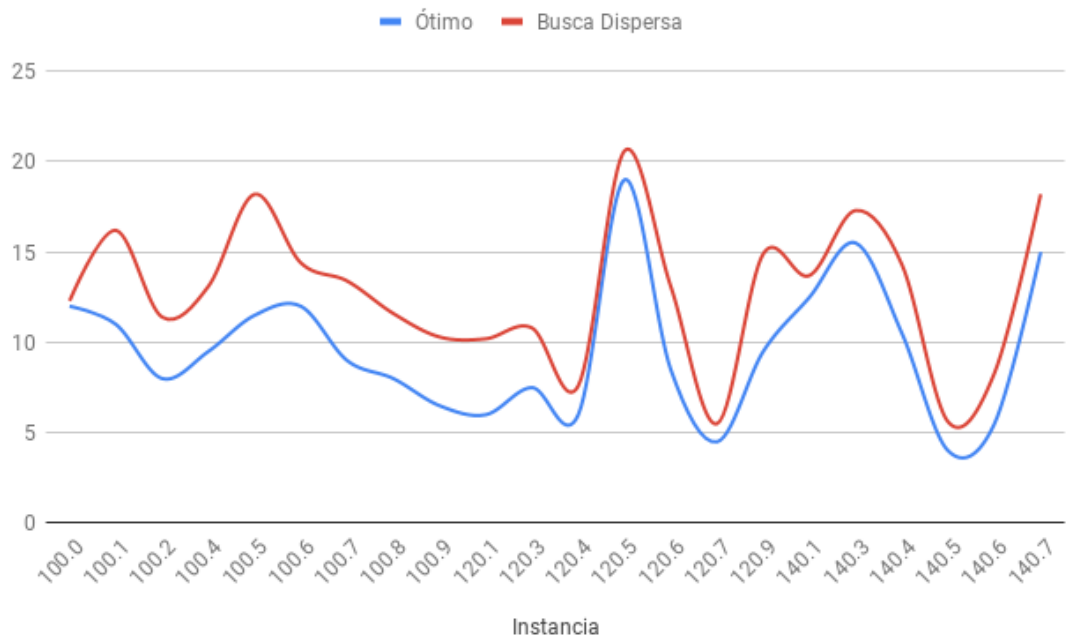


Figura 6 – Gráfico comparando o valor ótimo retornado pelo algoritmo exato e o valor obtido pela heurística nas instâncias de tamanho 100 até 140

Para as instâncias do Grupo 1, demonstradas nas tabelas 1 e **??**, o tempo de execução médio foi de $2,51$ segundos, enquanto o algoritmo exato demora em média $5,47$ segundos. Em relação aos valores das soluções, a heurística apresenta resultados aproximados do algoritmo exato em 13 instâncias (resultados em negrito na tabela), sendo o GAP médio, das 39 instâncias, igual a $0,37(37\%)$. Pode-se notar também que existem valores de GAP muito altos, chegando até a 70%.

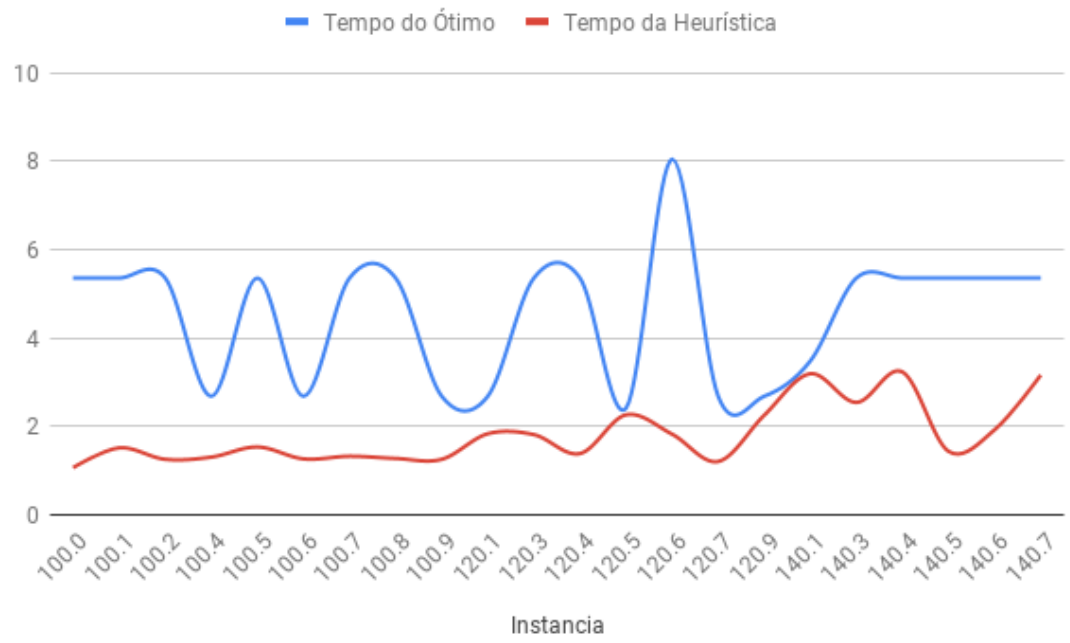


Figura 7 – Gráfico comparando o tempo do algoritmo exato e o tempo da heurística nas instâncias de tamanho 100 até 140

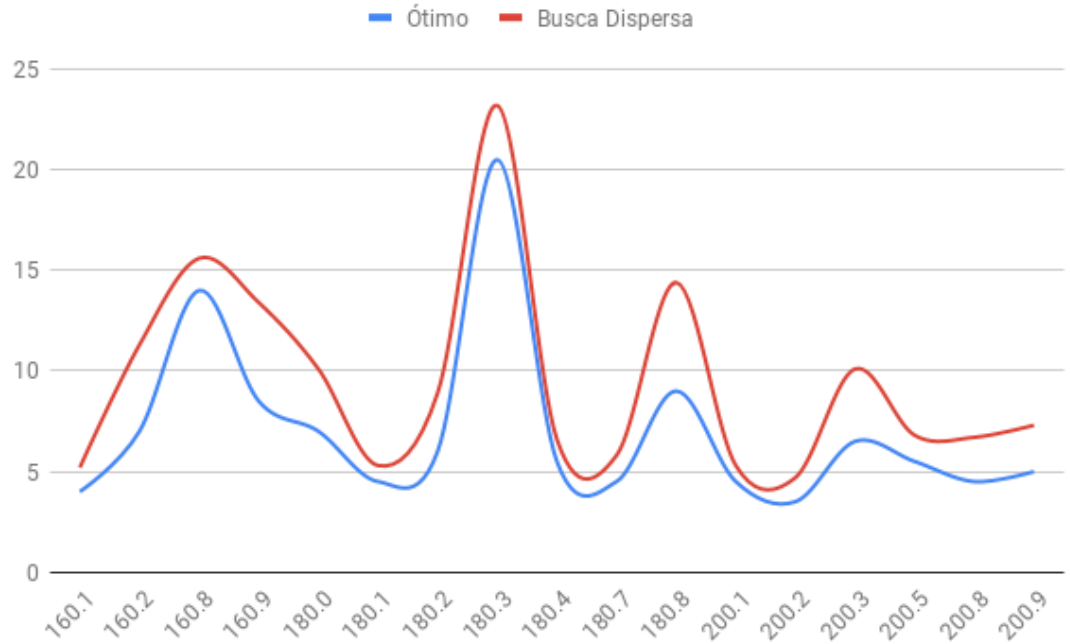


Figura 8 – Gráfico comparando o valor ótimo retornado pelo algoritmo exato e o valor obtido pela heurística nas instâncias de tamanho 160 até 200

Figura 9 – Gráfico comparando o tempo do algoritmo exato e o tempo da heurística nas instâncias de tamanho 160 até 200

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho definimos uma versão do Problema do Compartilhamento de Veículos com Restrições Físicas e Sociais, permitindo determinados tipos de ciclos, sendo um incremento ao problema apresentado em Rodrigues (2018). Para sua resolução propomos uma heurística baseada no algoritmo de Busca Dispersa. Esta solução pode ser utilizada para evitar vários problemas ambientais e de locomoção.

Para efetuar testes computacionais foram utilizadas as instâncias criadas por Rodrigues (2018) e instâncias geradas pelos autores. A análise dos resultados foi realizada através do cálculo do GAP (porcentagem entre o valor da solução heurística e o valor da solução ótima), onde verificou-se que, os resultados obtidos pela heurística são, em alguns casos, próximos do ótimo, porém ainda se faz necessário ajustes de parâmetros, para que este possa superar as demais instâncias onde o *GAP* ainda se apresenta muito alto. Em relação ao tempo de execução, a heurística demonstrou um tempo inferior comparado ao método exato, cerca de metade do tempo, entretanto, para ser mais preciso nos experimentos computacionais, se faz necessário o uso de instâncias mais desafiadoras, uma vez que as instâncias utilizadas para teste são de fácil resolução e portanto inviabilizam melhores análises de desempenho de algoritmos meta-heurísticos.

Para trabalhos futuros espera-se buscar outras formas de abordagem do algoritmo de Busca Dispersa para o problema, procurando outras maneiras de gerar soluções iniciais, ajustes de parâmetros, ou ainda, outras formas de abordagem do problema como um todo, como por exemplo, aplicação de outras heurísticas.

REFERÊNCIAS

ANDRADE, R. C. de. New formulations for the elementary shortest-path problem visiting a given set of nodes. **European Journal of Operational Research**, Elsevier, v. 254, n. 3, p. 755–768, 2016.

ARAÚJO, G. F. de. **Algoritmos Meta-heurísticos Para a solução do Problema do Caxeiro Viajante com Múltiplas Caronas**. Dissertação (Mestrado em Sistemas e Computação) — Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, 2016.

ARROYO, J. E. C. *et al.* **Heurísticas e metaheurísticas para otimização combinatória multiobjetivo**. Dissertação (Tese de Doutorado em Engenharia Elétrica) — Departamento de Engenharia de Sistemas, Universidade Estadual de Campinas, Campinas, 2002.

BlaBlaCar. **Caronas simplificam sua viagem**. [s.l.]: BLABLACAR, 2018. Disponível em: <https://www.blablacar.com.br/>. Accessado: 28/05/2018.

Carona Fácil. **Compartilhe Roteiros**. [s.l.]: CARONAFACIL, 2018. Disponível em: <https://www.caronafacil.com/>. Accessado: 28/05/2018.

COSTA, M. d. S. **Mobilidade urbana sustentável: um estudo comparativo e as bases de um sistema de gestão para Brasil e Portugal**. Dissertação (Mestrado em Engenharia Civil) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2003.

GLOVER, F. Heuristics for integer programming using surrogate constraints. **Decision sciences**, Wiley Online Library, v. 8, n. 1, p. 156–166, 1977.

LIMA, A. A. d. **Análise e Implementação de um Algoritmo de Busca Dispersa para o Planejamento da Expansão de Sistemas de Transmissão**. Dissertação (Mestrado em Engenharia Elétrica) — Universidade Estadual Paulista Júlio de Mesquita Filho, Faculdade de Engenharia de Ilha Solteira, 2012.

MÉNDEZ, Y.; GUARDIA, L. **Problema do caminho mais curto-Algoritmo de Dijkstra**. Rio de Janeiro: SPOLM, 2008.

NETO, A. o. D.; ANDRADE, L. D. de. Sistema de caronas intermunicipal tendo a cidade de florianópolis como centro. 2016.

RODRIGUES, D. N. **Problema de Compartilhamento de Veículos com Restrições Físicas e Sociais**. Monografia (Graduação em Engenharia de Software) — Campus Russas, Universidade Federal do Ceará, Russas, 2018.

SOSA, N. G. M.; GALVÃO, R. D.; GANDELMAN, D. A. Algoritmo de busca dispersa aplicado ao problema clássico de roteamento de veículos. **Pesquisa Operacional**, SciELO Brasil, v. 27, n. 2, p. 293–310, 2007.