

O problema de Roteamento de Veículos: Transporte de pessoas

Samuel Ferreira Guimarães Santos

Instituto de Ciências de Matemática e Computação, Universidade de São Paulo
samuel.ferreira.santos@usp.br

Orientadores: Franklina M. B. Toledo e Pedro Belin Castellucci

Instituto de Ciências de Matemática e Computação, Universidade de São Paulo
fran@icmc.usp.br, pedro.castellucci@ufsc.br

RESUMO

O Problema de Roteamento de Veículos (PRV) é um dos problemas mais estudados na área de otimização combinatória. Seu objetivo é planejar um conjunto ideal de rotas para uma frota de veículos para atender um conjunto de clientes. Neste trabalho, foi estudado o caso do PRV para o transporte de pessoas com embarque e desembarque simultâneo. O trabalho foi inspirado em um problema de roteirização da Associação de Pais e Amigos dos Excepcionais (APAE). A solução está baseada em uma implementação eficiente de um algoritmo genético. Foram realizados experimentos utilizando instâncias da literatura e baseada em cenários reais que mostraram a competitividade do método. Inclusive, atualizando alguns dos melhores resultados conhecidos na literatura. Além disso, durante o trabalho foi desenvolvida uma aplicação web. Os algoritmos, estrutura de dados e aplicação web foram totalmente desenvolvidos pelo autor.

PALAVRAS CHAVE. Meta-heurísticas, Problema do Roteamento de Veículos, Algoritmo Genético.

PM; L&T

ABSTRACT

The Vehicle Routing Problem (VRP) is one of the most studied problems in the combinatorial optimization field. Its goal is to plan an optimal set of routes for a vehicle fleet to serve a set of clients. In this work, we studied the VRP with simultaneous pickup and delivery. The work was inspired by a routing problem of the “Associação de Pais e Amigos dos Excepcionais (APAE)”. The solution is based on an efficient implementation of a genetic algorithm. Computational experiments using literature instances and random instances based on real data show that the method is competitive, even updating some of the best values known in the literature. Moreover, a web application was created in this work. The author developed the algorithms, data structures, and web application.

KEYWORDS. Metaheuristic, Vehicle Routing Problem, Genetic Algorithm.

PM; L&T

1. Introdução

Dantzig e Ramser [1959] foram os primeiros a definir o problema de roteamento de veículos que consiste em atender as demandas de um conjunto de clientes a partir de uma frota de veículos. De acordo com a literatura, existem muitas variantes do problema de roteamento de veículos [Golden et al., 2008]. Dentre elas, o problema de embarque e desembarque tem recebido muita atenção nas últimas décadas. O problema consiste em definir rotas para tratar o embarque e o desembarque de pessoas (ou a coleta e a entrega de mercadorias). Revisões sobre o tema podem ser encontradas em [Parragh et al., 2008a] e [Parragh et al., 2008b]

Esse trabalho foi inspirado no problema de definição de rotas para uma Associação dos Pais e Amigos dos Excepcionais (APAE) do estado de São Paulo. A instituição é uma Organização da Sociedade Civil, sem fins lucrativos, que promove a saúde das pessoas com deficiência intelectual, apoia sua inclusão social, incide na defesa de seus direitos, produzindo e disseminando conhecimento.

Atualmente, a APAE de São Carlos atende aproximadamente 400 pessoas divididas nos períodos da manhã e tarde. Podemos caracterizar o transporte dos alunos, como duas variantes de problemas de roteamento de veículos, uma para o período da manhã e para o final da tarde e outra para o período entre o fim da manhã e o início da tarde. Neste trabalho, focamos no segundo problema, em que os alunos da manhã são levados para casa e os alunos do período da tarde são trazidos para a APAE. Esse transporte pode ser definido como um Problema do Roteamento de Veículos com Embarque e Desembarque. Os outros dois podem ser considerados como subproblemas em que há somente o embarque ou desembarque dos alunos.

Esse problema pode ser definido como um grafo $G = (V, A)$, em que V é o conjunto de vértices e A o conjunto de arcos. O conjunto V é definido como $V = \{0\} \cup U$, onde o vértice 0 é a escola e U é o conjunto da localização dos estudantes. O conjunto de arcos é definido como $A = \{(i, j) : i, j \in V, i \neq j\}$ e a cada arco $(i, j) \in A$ é associado um valor não-negativo c_{ij} representando o custo de viagem entre os vértices i e j . As rotas começam no vértice 0, visitam um subconjunto de localização dos estudantes e terminam visitando o vértice 0 novamente. A restrição de capacidade dos veículos deve ser respeitada durante toda a rota. Cada estudante possui um par de requisição de pontos, um referente ao embarque e outro ao desembarque, sendo que estes são a residência do estudante e a localização da escola. O objetivo é encontrar um conjunto de rotas que minimize os custos de transporte dos estudantes. Como este é um problema de difícil resolução, desenvolvemos o método heurístico descrito a seguir para tratar o problema.

2. Método de solução desenvolvido

Para resolver o problema foram implementadas duas heurísticas construtivas: i) algoritmo de economias; e ii) algoritmo de varredura e quatro meta-heurísticas: VND, GRASP, ILS e Algoritmo Genético. Nesse trabalho, focaremos no Algoritmo Genético, que foi a meta-heurística que apresentou o melhor resultado. Os outros métodos podem ser encontrados em [Santos, 2019].

Algoritmos Genéticos são algoritmos de otimização global, baseados nos mecanismos de seleção natural e da genética ([Holland, 1975] e [Goldberg, 2006]). Em geral, um algoritmo genético envolve uma população de soluções, ou cromossomos, onde cada cromossomo codifica uma solução para uma instância em particular. A evolução ocorre através de operadores genéticos que replicam o fenômeno observado na natureza. Os operadores mais conhecidos são: cruzamento, mutação e seleção.

A seguir, descrevemos a representação de uma solução para o PRV, na forma de cromossomo e os operadores genéticos utilizados nesse trabalho. Eles foram baseados nos artigos de [Prins, 2004] e [Baker e Ayechew, 2003].

2.1. Representação do cromossomo

Um cromossomo é uma sequência (permutação) S de n pessoas atendidas, sem os delimitadores de rota. Por exemplo, se tivermos uma solução com duas rotas: $r_1 = (0, 4, 2, 0)$ e $r_2 = (0, 1, 3, 0)$, o cromossomo que representa essa solução é $(4, 2, 1, 3)$. Essa representação pode ser interpretada como a ordem em que as pessoas são visitadas, se apenas um veículo fosse responsável pelo transporte.

2.2. Função de *fitness*

Para avaliar o valor da função objetivo do cromossomo, é preciso dividi-lo em rotas. Em [Prins, 2004] é proposto um procedimento de divisão ótimo. Considerando que o cromossomo é uma lista de valores inteiros, no intervalo entre 1 e n . O valor ótimo pode ser encontrado pela seguinte relação de recorrência:

$$dp_j = \begin{cases} 0, & \text{se } j = 0 \\ \min_{i=1}^{i=j} (dp_{i-1} + \text{custo da rota formada pelos clientes entre } i \text{ e } j), & \text{c.c.} \end{cases}$$

A solução ótima é dada por dp_n . O algoritmo de [Prins, 2004] tem complexidade de tempo $O(n^2)$. Para este trabalho, reduzimos a complexidade do método de divisão ótima para $O(n)$, os detalhes podem ser encontrados em [Santos, 2019]. Isso nos permitiu avaliar um maior número de gerações com o mesmo tempo de execução, se comparado ao método proposto em [Prins, 2004].

2.3. Cruzamento

Os pais utilizados para a reprodução são selecionados através do método do torneio. Para cada pai, no torneio, um número k de indivíduos são escolhidos aleatoriamente da população, e entre esses, o que tiver o menor valor de função *fitness* é selecionado. Como em [Baker e Ayeche, 2003], foi implementado o método com $k = 2$.

Como destacado por [Prins, 2004], ao representarmos o cromossomo sem delimitadores, podemos utilizar os operadores de cruzamento clássicos da literatura para o Problema do Caixeiro Viajante. Dentre eles, se destaca o cruzamento OX (*order crossover*). Esse processo pode ser implementado com complexidade $O(n)$.

2.4. Mutação

No método desenvolvido, um filho C produzido pelo cruzamento OX é melhorado por busca local com uma dada probabilidade p_m . O cromossomo C é convertido para uma solução do PRV, aplicando o algoritmo de divisão, explicado na função *fitness*. Por fim, é feita uma busca local na solução utilizando seis vizinhanças: três intra-rotas e três inter-rotas. São elas: 2-Opt, 3-Opt, Or-Opt, relocação, intercâmbio e cruzamento.

2.5. População inicial

A população do algoritmo é iniciada com: dois indivíduos dados pelas duas soluções das heurísticas construtivas de [Clarke e Wright, 1964] e de [Gillett e Miller, 1974], p indivíduos gerados utilizando uma heurística construtiva pseudo-gulosa baseada no algoritmo de [Clarke e Wright, 1964], e os demais indivíduos são gerados aleatoriamente. Destacamos que p é uma porcentagem do tamanho da população.

2.6. Diversificação da População

Um método simples de controle de diversidade foi implementado para impedir que dois cromossomos iguais estejam presentes na população. Além disso, também foi utilizado um método para reinicialização da população. O procedimento implementado foi baseado em [Cheung et al., 2001] para o problema de localização de facilidades, e é chamado de procedimento de substituição parcial. [Prins, 2004], utilizou esse método para o PRV e cita que experimentos numéricos mostram que, comparado com o método tradicional, esse produz uma descida mais rápida da função objetivo.

2.7. Algoritmo Genético Completo

O Algoritmo 1 descreve o algoritmo genético completo. Nas linhas 1 a 2, a população inicial é construída e ordenada pela função *fitness*, e o contador de iterações sem melhoria é inicializado. Em 5 e 6, é feita a seleção e cruzamento de duas soluções. Na linha 8, é realizada a permutação com probabilidade p_m de ocorrer. Em 10 e 11, é feita a troca de um indivíduo com valor de função objetivo acima da mediana pelo cromossomo recém-criado. Nas linhas 12-16, é atualizado o valor de β , checando se o indivíduo C é a nova melhor solução. Por fim, se o contador de soluções sem melhoria alcançar seu valor máximo, é iniciado o procedimento de substituição parcial da população.

Algoritmo 1: Pseudocódigo Algoritmo Genético

```
1 Inicializa população  $P$ , com indivíduos aleatórios e heurísticas construtivas.
2 Avaliação e ordenação de  $P$ .
3  $\beta = 0$ .
4 para  $k = 1, \dots, \text{Max-Gerações}$  faça
5     Selecione dois indivíduos  $P_1$  e  $P_2$ , pelo método do torneio de dois.
6     Aplique o cruzamento OX para  $(P_1, P_2)$  e selecione um filho  $C$  aleatoriamente.
7     se  $\text{random} < p_m$  então
8         | Aplique busca local no filho  $C$ .
9     Selecione uma posição  $k$  aleatoriamente no intervalo  $[\text{floor}(|P|/2), |P|]$ .
10    Remova o indivíduo na posição  $k$ . Insira  $C$  em  $P$  de forma ordenada.
11    se o filho  $C$  for igual a  $P_0$  então
12        |  $\beta = 0$ .
13    senão
14        |  $\beta = \beta + 1$ .
15    se  $\beta = \beta_{\text{max}}$  então
16        | Aplique o procedimento de substituição parcial.
```

3. Testes Computacionais

Os métodos foram implementados em C++ e compilados utilizando o g++ versão 5.4.0 e com a *flag* de otimização -O3. O ambiente computacional era equipado com um processador Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, memória RAM de 16 GB e o sistema operacional Ubuntu 16.04.

3.1. Definição das instâncias

Foram realizados testes computacionais utilizando dois conjuntos de instâncias: i) instâncias da literatura (Van Breedam [1994] e Taillard [1993]), e ii) instâncias geradas com base em dados reais. Por questões burocráticas não foi possível ter acesso a dados reais da APAE São Carlos, logo geramos instâncias com base em endereços obtidos aleatoriamente para as cidades de São Carlos e de São Paulo. A seguir detalhamos as instâncias utilizadas.

3.2. Instâncias da literatura

Utilizamos dois conjuntos clássicos de instâncias da literatura: o benchmark de Van Breedam [1994] e o de Taillard [1993]. As instâncias de Van Breedam são caracterizadas pela distribuição de clientes, localização dos depósitos, janelas de tempo, coleta e entrega, e demandas homogêneas ou heterogêneas. O último trabalho encontrado que reportou resultados utilizando esse benchmark foi em [Frutos et al., 2016]. A vantagem de utilizar esse benchmark é que algumas instâncias se assemelham como o transporte do período entre manhã e tarde, contendo pontos de embarque e desembarque. Porém, a otimalidade das soluções ainda não foram comprovadas. Por isso, também

utilizamos o benchmark de Taillard, que é composto por 12 instâncias, onde os clientes estão localizados num plano, espelhados em diversos agrupamentos. Para cada uma das instâncias o valor ótimo é conhecido, e isso, nos permite testar nosso método em relação a otimalidade.

3.3. Instâncias baseadas em dados reais

Neste estudo, para simular os dados de uma APAE, geramos instâncias com base em endereços de duas cidades, São Carlos e São Paulo. Desta forma, poderíamos avaliar o método proposto para instâncias da nossa cidade e uma cidade de grande. Foram geradas três instâncias para cada uma das cidades, nomeadas da seguinte forma: Sao-Paulo- n e Sao-Carlos- n , onde n representa a quantidade de pontos.

3.3.1. Instâncias para a cidade de São Carlos

O procedimento adotado foi fixar a APAE em seu endereço real. As coordenadas correspondentes aos endereços dos alunos foram geradas uniformemente distribuídas ao redor da APAE com um raio de 7 km da APAE. Definimos a distância de 7 km com base em dados do *Google Maps*, que estima que um veículo demora 20 minutos para percorrer 7 km dentro da cidade de São Paulo. Como o tempo limite para embarque dos alunos é 1 hora, estimamos que teríamos 20 minutos para nos distanciar da APAE, 20 minutos para retornar e 20 minutos para o restante do trajeto.

Após a obtenção das coordenadas geográficas, avaliamos se o ponto gerado pode representar um endereço físico ou comercial da cidade. Essa avaliação é realizada enviando o ponto gerado para um Sistema de Informação Geográfico (SIG). Na Figura 1, ilustramos uma instância gerada para a cidade de São Carlos.

Uma vantagem desse conjunto de instâncias é que representa endereços reais de São Carlos, com distância e tempo de viagem obtidos utilizando o SIG para um percurso utilizando um veículo, comumente um carro popular. Por outro lado, não consideramos a densidade demográfica, ao gerar pontos uniformemente distribuídos. Logo, criamos um segundo conjunto de instâncias baseados em endereços reais da cidade de São Paulo.

Figura 1: Exemplo de instâncias para São Carlos (a esquerda) e São Paulo (a direita).



3.3.2. Instâncias para a cidade de São Paulo

Seguindo a ideia apresentada em [Sartori, 2019], utilizamos o *OpenAddresses*, para obter endereços reais da cidade de São Paulo. Esta mesma base não foi utilizada para a cidade de São Carlos por ter poucas entradas para cidade. No entanto, para a cidade de São Paulo são encontradas 3.793.394 entradas.

A geração dos endereços foi realizada através de uma amostra de n locais respeitando os critérios já apresentados na Subseção 3.3.1. Note que, ao selecionarmos pontos dessa base de dados, temos uma aproximação da distribuição urbana, já que, regiões com maior densidade demográfica possuem mais endereços na lista e maior probabilidade de serem selecionados. Contudo, não é uma aproximação perfeita, porque depende dos dados disponíveis na base. Um exemplo de uma instância gerada é apresentado na Figura 1.

3.4. Resultados computacionais

Cada instância foi resolvida 20 vezes, com um tempo limite de um minuto de execução. Foi utilizado o IRACE para realizar a calibragem do número de indivíduos por população e da probabilidade de mutação do AG, os seguintes valores foram utilizados i) número de indivíduos por população: 25, 50, 75 e 100; ii) probabilidade de mutação: 10%, 15%, 20%, 25%, 30%, 35% e 40%. Os melhores resultados foram obtidos utilizando 50 indivíduos por população e probabilidade de mutação de 20%.

Na Tabela 1, na primeira coluna é descrita a instância resolvida, em seguida, é apresentado o valor da melhor solução conhecida da literatura (*Sol_Lit*) (reportado em Frutos et al. [2016]). Nas colunas seguintes, são apresentados, respectivamente, o valor da melhor solução encontrada (*Sol_AG*) e o desvio percentual desta solução em relação a literatura. Por fim, nas última coluna são apresentadas a solução média das 20 rodadas dos métodos com seu desvio padrão, a mediana e o desvio percentual da mediana em relação a literatura.

Tabela 1: Comparação do algoritmo genético com a literatura.

Instância	<i>Sol_Lit</i>	<i>Sol_AG</i>	GAP (%)	Média	Mediana	GAP Mediana (%)
Bre-1	1106,0	1106,0	0,00	1106,50 $\pm$ 0, 53	1106,5	0,05
Bre-2	1506,0	1506,0	0,00	1506,00 $\pm$ 0, 00	1506,0	0,00
Bre-3	1751,0	1751,0	0,00	1751,00 $\pm$ 0, 00	1751,0	0,00
Bre-4	1470,0	1470,0	0,00	1470,00 $\pm$ 0, 00	1470,0	0,00
Bre-5	950,0	951,0	0,11	952,90 $\pm$ 1, 20	953,0	0,32
Bre-6	969,0	969,0	0,00	971,30 $\pm$ 1, 42	971,5	0,26
Bre-7	1032,0	1016,0	-1,55	1031,40 $\pm$ 8, 85	1032,0	0,00
Bre-8	1067,0	1032,0	-3,28	1034,40 $\pm$ 2, 63	1034,0	-3,09
Bre-9	1690,0	1691,0	0,06	1694,40 $\pm$ 3, 78	1693,0	0,18
Bre-10	1026,0	1026,0	0,00	1027,70 $\pm$ 1, 77	1027,5	0,15
Bre-11	1128,0	1124,0	-0,35	1126,10 $\pm$ 2, 42	1125,0	-0,27
GAP Médio			-0,45			-0,22

Das onze instâncias avaliadas, para seis foram obtidas soluções com o mesmo custo reportado na literatura. Em três casos, foram encontradas rotas com custo melhor do que reportado na literatura. Essas rotas estão detalhadas em [Santos, 2019]. Como podemos observar, os resultados mostram que o método é robusto e teve melhor desempenho em média que a literatura (-0,45%).

Na Tabela 2, são apresentados os resultados obtidos resolvendo as instâncias do benchmark de Taillard. A tabela traz os mesmos dados da Tabela 1, exceto pela segunda coluna, em que o valor da solução ótima é apresentado. Para este conjunto de instâncias, o algoritmo genético mais uma vez apresentou uma boa performance, obtendo um desvio percentual médio de 0,15% da solução ótima. Para cinco das doze instâncias, o método obteve a solução ótima do problema.

Na Tabela 3, reportamos os resultados obtidos para as instâncias geradas. A primeira coluna indica a instância, nas colunas seguintes, temos, respectivamente, a melhor solução inicial encontrada (*Sol_Ini*), a melhor solução obtida pelo algoritmo genético (*Sol_GA*) e o tempo que o GA demandou (*T_Sol*). Na última coluna, apresentamos a melhoria percentual entre a solução inicial e a melhor solução encontrada pelo GA ($Me_{Si_GA=100} = \frac{Sol_Ini - Sol_GA}{Sol_Ini}$).

Embora não tenhamos a solução ótima para as instâncias geradas, observamos que as heurísticas construtivas fornecem boas soluções, quando levamos em consideração o tempo consumido (todas as instâncias encontraram solução em menos de um segundo), no entanto, o algo-

ritmo genético conseguiu melhorar significativamente a qualidade das soluções iniciais (21,17% em média) com um tempo inferior a 1 minuto.

Tabela 2: Resultados do algoritmo genético para o benchmark de Taillard.

Instância	<i>Sol_Otima</i>	<i>Sol_GA</i>	GAP (%)	Média	Mediana	GAP Mediana (%)
tai75a	1618,36	1618,36	0,00	1618,36 $\pm$ 0,00	1618,36	0,00
tai75b	1344,64	1344,64	0,00	1344,64 $\pm$ 0,00	1344,64	0,00
tai75c	1291,01	1291,01	0,00	1291,01 $\pm$ 0,00	1291,01	0,00
tai75d	1365,42	1365,42	0,00	1365,42 $\pm$ 0,00	1365,42	0,00
tai100a	2041,34	2047,90	0,32	2049,92 $\pm$ 2,05	2048,77	0,36
tai100b	1939,90	1940,61	0,04	1941,22 $\pm$ 0,98	1940,61	0,04
tai100c	1406,20	1406,20	0,00	1407,25 $\pm$ 1,04	1407,29	0,08
tai100d	1580,46	1586,08	0,36	1589,03 $\pm$ 3,26	1586,97	0,41
tai150a	3055,23	3056,85	0,05	3062,92 $\pm$ 5,37	3063,38	0,27
tai150b	2727,23	2732,19	0,18	2734,26 $\pm$ 2,69	2732,51	0,19
tai150c	2358,66	2362,91	0,18	2369,35 $\pm$ 6,67	2365,60	0,29
tai150d	2645,39	2659,12	0,52	2662,78 $\pm$ 4,16	2663,21	0,67
GAP Médio			0,14			0,19

Tabela 3: Desempenho do Algoritmo Genético nas instâncias baseadas em dados reais.

Instância	<i>Sol_Ini(km)</i>	<i>Sol_GA(km)</i>	<i>T_Sol(s)</i>	<i>Mel_{Sol_GA}</i>
SaoPaulo-100	227,62	167,04	49,95	26,61
SaoPaulo-200	392,67	303,36	36,52	22,74
SaoPaulo-400	623,47	522,75	12,97	16,15
SaoCarlos-100	208,31	144,20	25,21	30,78
SaoCarlos-200	331,16	283,20	58,53	14,48
SaoCarlos-400	494,00	413,68	44,37	16,26
GAP Médio				21,17

4. Conclusão

Neste trabalho, foi estudado o problema de transporte de pessoas, mais especificamente o estudo foi inspirado na definição de rotas para uma Associação de Pais e Amigos dos Excepcionais (APAE). Um algoritmo genético foi desenvolvido para sua resolução e foi avaliado com instâncias da literatura e instâncias geradas com base em dados reais. Os resultados obtidos apontam que o método é promissor, encontrando soluções melhores que a literatura para parte das instâncias avaliadas. Testes com mais instâncias devem ser realizados em trabalhos futuros. Para as instâncias baseadas em dados reais, as heurísticas construtivas forneceram boas soluções com baixo tempo de processamento. Porém, se o usuário priorizar a qualidade da função objetivo, o algoritmo genético é indicado. Os algoritmos apresentados foram incorporados em uma aplicação web, também desenvolvida pelo autor, disponível em: <http://logisticasocial.lot.icmc.usp.br/> (usuário: sbpo senha: sbpo2020!).

Agradecimentos

Esse trabalho foi desenvolvido com apoio financeiro da FAPESP (18/06974-0) e CEPID-CeMEAI FAPESP (13/07375-0). Agradecemos aos revisores por colaborarem com a melhoria deste trabalho.

Referências

- Baker, B. M. e Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30:787 – 800.
- Cheung, B.-S., Langevin, A., e Villeneuve, B. (2001). High performing evolutionary techniques for solving complex location problems in industrial system design. *Journal of Intelligent Manufacturing*, 12:455–466.
- Clarke, G. e Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581.
- Dantzig, G. B. e Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6: 80–91.
- Frutos, M., Tohmé, F., e Miguel, F. (2016). A genetic algorithm's approach to the optimization of capacitated vehicle routing problems. In Vasant, P., Weber, G.-W., e Dieu, V. N., editors, *Handbook of Research on Modern Optimization Algorithms and Applications in Engineering and Economics*, p. 209–235. Hershey, PA: IGI Global.
- Gillett, B. E. e Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22:340–349.
- Goldberg, D. (2006). *Genetic Algorithms*. Pearson Education.
- Golden, B. L., Raghavan, S., e Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media.
- Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.
- Parragh, S., Doerner, K., e Hartl, R. (2008a). A survey on pickup and delivery problems: Part i: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58:21–51.
- Parragh, S., Doerner, K., e Hartl, R. (2008b). A survey on pickup and delivery problems: Part ii: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, 58: 81–117.
- Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31:1985 – 2002.
- Santos, S. F. G. (2019). Logística social: o transporte de pessoas. Trabalho de Conclusão de Curso. ICMC, Universidade de São Paulo. <https://drive.google.com/file/d/1SdT3lpmax8Kukrh1ky5W17ODaMNgtEID/view?usp=sharing>. Acesso: 15/09/2020.
- Sartori, C. S. (2019). The pickup and delivery problem with time windows: algorithms, instances, and solutions. Master's thesis, Universidade Federal do Rio Grande do Sul.
- Taillard, (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8): 661–673.
- Van Breedam, A. (1994). *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints*. PhD thesis.