Heurísticas e metaheurísticas aplicadas ao problema de transporte compartilhado com atendimento suficientemente próximo

Bruno Soares Santos

JUIZ DE FORA

NOVEMBRO, 2017

Heurísticas e metaheurísticas aplicadas ao problema de transporte compartilhado com atendimento suficientemente próximo

Bruno Soares Santos

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Carlos Cristiano Hasenclever Borges

Co-orientador: Luciana Brugiolo Gonlçaves

JUIZ DE FORA

NOVEMBRO, 2017

Heurísticas e metaheurísticas aplicadas ao problema de transporte compartilhado com atendimento suficientemente próximo

Bruno Soares Santos

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Carlos Cristiano Hasenclever Borges
Doutor em Engenharia Civíl

Stênio Sã Rosario Furtado Soares
Doutor em Computação

Raul Fonseca Neto
Doutor em Engenharia de Sistemas e Computação

JUIZ DE FORA
15 DE NOVEMBRO, 2017

Resumo

Transporte compartilhado envolve principalmente o compartilhamento de veículos entre pessoas a fim de facilitar o deslocamento urbano, e é um grande reforço ao combate aos problemas enfrentados pelo aumento da quantidade de veículos em circulação em ambientes urbanos, como aumento da poluição, estresse e perda de tempo no trânsito, acidentes e demanda de estacionamentos.

Este trabalho apresenta uma formulação para uma variação do problema de transporte compartilhado, abordando uma situação semelhante a realidade de empresas e escolas. Neste problema, um grupo de motoristas dispostos a sair de seu trajeto para oferecer carona, desde que o desvio feito não seja muito grande, e um grupo de passageiros que apresentam um conjunto de pontos próximos a seus pontos de origem onde os carros poderiam busca-los, considerando o possível deslocamento do passageiro a fim de facilitar a rota do motorista. Foi abordado apenas o cenário considerando-se um destino comum à todos.

O objetivo foi desenvolver um modelo para o problema e algoritmos capazes alocar o máximo de passageiros a carros e, se possível, reduzir o trajeto total percorrido pelos carros.

Foram apresentadas duas heurísticas para a solução do problema assim como simplificações no modelo de uma cidade real a fim de facilitar a implementação e realização dos testes. Este trabalho considera uma mudança de paradigma em relação ao que normalmente se encontra na literatura, revertendo a abordagem do problema que geralmente é resolvido de maneira a tentar encontrar a melhor rota para os carros, e passando para uma tentativa do ponto de vista do passageiro, ao se tentar encontrar o carro que melhor atenda cada passageiro.

Palavras-chave: Grafos, Guloso, GRASP, Algoritmo Genético, Heurísticas, Roteamento de Veículos, Entrega e Coleta, Transporte Compartilhado, Carona.

Abstract

Shared transportation involves mainly the sharing of vehicles between people in order to facilitate urban mobility, and is a great reinforcement to combat the problems faced by the increase of the number of vehicles circulating in urban environments, such as increased pollution, stress and loss of time in traffic jams, accidents and demand for parking lots.

This work presents a formulation for a variation of the problem in the area of shared transportation, addressing a situation similar to the reality of companies and schools. In this problem, we have a group of drivers willing to deviate out of their way to offer a ride, as long as the diversion is not very large, and a group of passengers that present a set of points near their origins where cars could fetch them, considering the possibility of the passageger to dislocate, making a better route possible. It was only considered a scenario where the destination is the same for all drivers and passengers.

The objective was to develop a model for the problem and algorithms capable of allocating the maximum number of passengers to the cars, if possible, reducing the total distance traveled by the cars.

Two heuristics were presented for the solution of the problem as well as simplifications in the model of a real city in order to facilitate the implementation and realization of the tests. This work considers a change of paradigm in relation to what is usually found in the literature, reversing the approach of the problem that is usually solved in order to try to find the best route for the cars, and turning to an attempt from the point of view of the passenger, when trying to find the car that best attend each passenger.

Keywords: Graph, Greedy, Genetic Algorithm, Heuristic, Shared Transportation, Pick up and Deliver, carpooling, Shared Transportation.

Agradecimentos

Gostaria de agradecer aos professores, Carlos, Luciana e Stênio, que me orientaram durante a realização deste projeto.

Aos meu pais pelo suporte, incentivo, carinho e sem os quais nada disso seria possível.

A todos meus amigos, com atenção especial aos Cabras Virgens, que me acompanharam durante todos estes anos, sendo a melhor companhia que eu poderia esperar.

A minha namorada Bell pela paciência, atenção e ajuda em todos os momentos que precisei.

"It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers."

Patrick Rothfuss, The Wise Man's Fear

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
AG	Algoritmo Genético
GRASP	*Greedy Randomized Adaptive Search Procedure*
mTSP	*Multiple Traveling Salesman Problem*
Darp	*Dial a ride problem*
OSM	*Open Street Maps*
XML	*eXtensible Markup Language*
ILS	*Iteraded Local Search*

1 Introdução

Problemas de otimização em grafos são amplamente estudados por cientistas da computação por permitirem uma fácil modelagem de problemas reais e apresentarem uma complexidade elevada.

Existem diversos problemas de roteamento de veículos onde são modelados sistemas viários na forma de grafos e programas procuram definir a rota de veículos visando minimizar alguns objetivos de interesse, como tempo ou custo de deslocamento, a distância percorrida, ou até maximizar uma função de obtenção de ganhos, como o atendimento de determinados pontos que oferecem diferentes recompensas.

Neste trabalho, abordamos um problema de transporte compartilhado ou *Ridesharing*, que nada mais é que o compartilhamento de meios de transporte por duas ou mais pessoas e tem como objetivo a melhor utilização dos recursos de transporte, economizando combustível e diminuindo a quantidade de veículos em circulação, melhorando o trânsito nas cidades.

1.1 Justificativa

De acordo com Netto and Ramos [2017], no período 2003 - 2012 a frota de automóveis nos municípios brasileiros com mais de 60 mil habitantes aumentou 70%, em contraste com o aumento de 16% da população no mesmo período. Neste artigo ainda são apresentados dados quanto a porcentagem dos gastos com transporte relacionado ao transporte privado (80%), e o aumento na porcentagem de trabalhadores que passam mais de uma hora no trânsito entre trabalho-casa (15% para 17%), entre outros indicadores. Logo, o aumento na quantidade de veículos nas cidades é, hoje em dia, um dos maiores problemas encontrados na mobilidade urbana, pois com a sobrecarga do sistema viário, a crescente formação de engarrafamentos gera o desperdício de tempo e combustível, uma maior demanda por estacionamentos, aumento de poluição ambiental e maior possibilidade de acidentes.

O transporte compartilhado apresenta uma alternativa para mitigar este pro-

blema, pois com motoristas compartilhando seus veículos com um ou mais passageiros, teremos menor necessidade de veículos circulando simultaneamente e um melhor atendimento aos usuários dos sistemas de transporte. Além da possibilidade da divisão dos custos com transporte.

Os problemas na computação relacionados a *ridesharing* apresentam aplicações e simulações de situações reais e têm por objetivo garantir que o maior número de pessoas chegue ao destino, otimizando os recursos utilizados. Neste trabalho, será abordado o problema considerando a restrição de atendimento próximo de passageiros proposto por Balardino and Santos [2016], que considera um caso próximo da realidade de empresas ou colégios, em que tem-se um único destino para todos os passageiros e motoristas, a possibilidade do desvio de motoristas de suas rotas a fim de atender um maior número de passageiros, e ainda a possibilidade de deslocamento do passageiro de sua origem, a fim de facilitar o trajeto do motorista. No capítulo 2 é mostrado como foi feita a modelagem para o problema.

1.2 Objetivos

Propor e avaliar algoritmos que retornem soluções viáveis dentro das restrições do problema proposto que obtenham bons resultados em relação ao número de passageiros atendidos e minimizando, se possível, os desvios dos carros.

Considerar o tempo de execução das heurísticas e sua escalabilidade, uma vez que uma aplicação real pode conter uma carga realmente grande de dados e, apesar de o problema ser primariamente estático, é interessante que se possa encontrar soluções rapidamente em casos de modificações inesperadas na entrada do problema, por imprevistos como por exemplo o defeito em um carro de um dos motoristas.

1.3 Organização do texto

O texto foi estruturado em 7 capítulos. O Capítulo 1 apresenta uma introdução sobre o problema abordado, os objetivos e motivação que levaram ao desenvolvimento deste trabalho. O Capítulo 2 formaliza o modelo utilizado para a criação de instâncias. O

Capítulo 3 apresenta a metodologia aplicada na resolução do problema apresentado. O Capítulo 4 contém os estudos realizados na literatura, mostrando problemas e formas de solução similares aos apresentados posteriormente. No Capítulo 6 são apresentados os parâmetros aplicados nos testes e os resultados obtidos. Finalmente o Capítulo 7 mostra as conclusões obtidas pelo autor e sugestões de trabalhos futuros.

2 Formalização do Problema

O problema de *Ridesharing* com atendimento suficientemente próximo abordado neste trabalho visa abordar as condições mais parecidas possíveis de uma situação real, logo foi modelada uma cidade na forma de um grafo, contendo um grupo de motoristas, um grupo de possíveis passageiros e um destino comum a todos. O objetivo é alocar o maior número de passageiros a carros de maneira que o desvio total da rota mínima entre carros e o destino seja mínimo.

Tendo em vista simplificar o modelo, todo destino, origem de motorista e pontos de atendimento dos passageiros foi associados aos vértices do grafo, estes que representam elementos das ruas, como por exemplo interseções entre vias e pontos finais de ruas sem saídas. Na Seção 6.1 é descrito como foi feita a geração das instâncias de teste.

Na modelagem do problema, considera-se que a cada passageiro é associado um conjunto de pontos possíveis de embarque, onde existe a possibilidade de o passageiro andar até um ponto mais conveniente para o motorista, dentro de um dado raio de deslocamento do seu ponto inicial. Outra especificidade do problema considerada foi um desvio máximo da rota mínima do motorista até o destino, uma vez que este pode não estar disposto a sair de sua rota para atender passageiros que estejam muito longe.

A Figura 2.1 apresenta um exemplo dos elementos descritos no problema, onde em amarelo se apresenta a origem de um passageiro com uma representação da distância de atendimento, onde os pontos dentro deste raio são considerados para o atendimento, e ainda, o exemplo de um carro que faz um desvio de sua rota mínima para o atendimento deste passageiro.

O modelo proposto se assimila ao utilizado por Balardino et al. [2016]. Contudo para os experimentos realizados neste trabalho foram feitas simplificações na modelagem do grafo das cidades a fim de facilitar a obtenção de novas instâncias e implementação dos algoritmos. Foi utilizada a matriz de distâncias mínimas entre todos os vértices para o cálculo das rotas. Portanto, todo nó do grafo é considerado como um possível ponto de retorno, o que nem sempre reflete uma situação real.

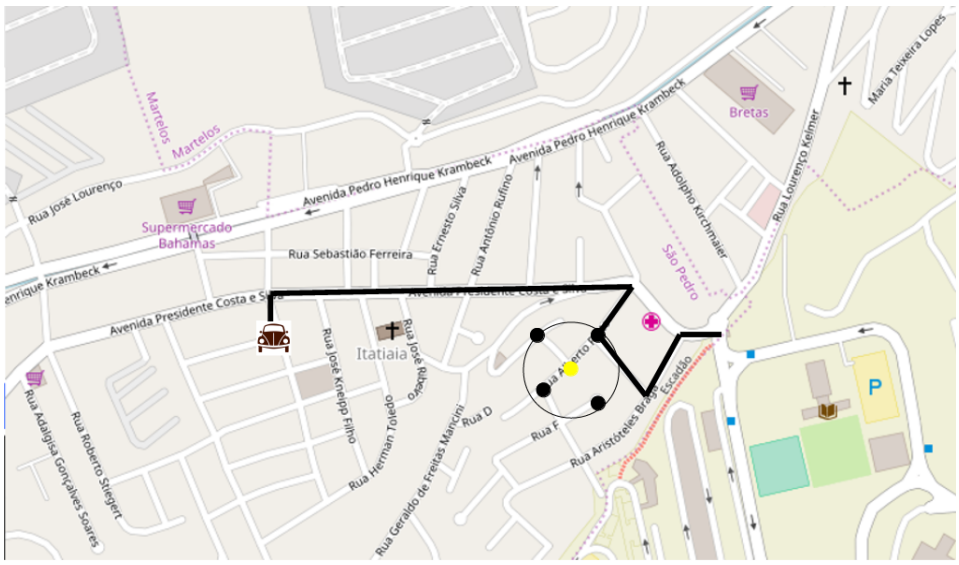

Figura 2.1: Exemplo dos elementos do problema

Dados de entrada:

- V - conjunto de vértices do grafo
- K - conjunto de motoristas
- P - conjunto de passageiros
- $M_{i,j} \in \mathbb{R}, i, j \in V$ - custo do caminho mínimo entre os vértices i, j
- $O_k \in V, k \in K$ - origem do motorista k
- $D \in V$ - destino de todos os motoristas e passageiros
- $D_{max} \in \mathbb{R}$ - porcentagem de desvio máximo
- $S_p \in V, p \in P$ - conjunto de pontos suficientemente próximo do passageiro p
- $L \in \mathbb{R}$ - bônus por atender um passageiro (valor suficientemente grande)

Variáveis:

- $a_k \in \mathbb{N}, k \in K$ - Número de passageiros atendidos pelo carro k
- $c_k \in V, k \in K$ - Vetor com posições, em ordem de visitação, dos pontos relevantes (origem, destino e atendimentos) percorridos por k, onde $c_k^i, i \in \mathbb{N}/2 \leq i \leq a_k + 2$, representa o *i-ésimo* vértice no caminho de k

- $y_{p,k}, p \in P, k \in K$ - variável binária, 1 caso o carro k atenda o passageiro p, 0 caso contrário

- $d_k \in \mathbb{R}, k \in K$ - desvio do carro k, calculado pela fórmula $(\sum_{i=1}^{at_k+1} M_{c_k^i, c_k^{i+1}}) - M_{O_k,D}$, ou seja, o percurso de k na solução, menos o trajeto mínimo entre a origem do carro k e o destino

Função objetivo:

$$Max\, Z = \sum_{p \in P} \sum_{k \in K} L\, y_{p,k} - \sum_{i \in K} d_i \tag{2.1}$$

Restrições:

$$M_{O_k,D} + d_k \leq D_{max}.M_{O_k,D}, \forall k \in K \tag{2.2}$$

$$0 \leq a_k \leq 4, \forall k \in K \tag{2.3}$$

$$\sum_{k \in K} y_{p,k} \leq 1, \forall p \in P \tag{2.4}$$

$$c_k^1 = O_k, \forall k \in K \tag{2.5}$$

$$c_k^{at_k+2} = D, \forall k \in K \tag{2.6}$$

$$y_{p,k} = 1 \rightarrow S_p \cap c_k \neq \emptyset \tag{2.7}$$

Assim a Equação 2.1 garante, pelo primeiro termo, que as melhores soluções serão aquelas que atenderem o maior número de passageiros, enquanto o segundo termo minimiza o desvio feitos pelos carros. A constante L deve ter valor suficientemente grande de forma que o primeiro termo tenha maior relevância, priorizando assim as soluções que atenderem ao maior número de passageiros. As Restrições 2.2 definem o desvio máximo de um carro como uma porcentagem de sua distância mínima até o destino, 2.3 define que,

para cada carro, o número de passageiros atendidos não deve ser maior que 4, 2.4 garante que cada passageiro será atendido por no máximo um único carro, 2.5 e 2.6 garantem que carro sempre partirá de sua origem e chegará ao destino. Por fim, a Equação 2.7 garante que um veículo só pode atender a um passageiro, se este estiver com pelo menos um ponto de atendimento em seu caminho.

A Figura 2.2 apresenta os dados de entrada de uma instância, onde a primeira linha contém as informações sobre o número de carros, de passageiros, o destino e o desvio máximo, em seguida tem-se as origens dos motoristas e uma linha para cada passageiro representando o número de pontos, seguido dos pontos. A Figura 2.3 mostra a matriz de distâncias minímas entre cada ponto do grafo, sendo a primeira linha o número de pontos, seguida da matriz $M_{nPts\,X\,nPts}$ com as distâncias mínimas.

```
20 25 0 1.5 ⬅ Nmotoristas nPassageiros Destino DesvioMáximo
892 1764 1162 2271 200 575 1682 2038 1381 2249 1870 1325 896 2177 854 219 1979 1471 1606 1424 ⬅ origemCarros
11 1492 1493 1494 1954 1955 1956 1957 1958 1959 1960 1961
43 601 602 613 614 615 616 617 618 619 620 621 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645
646 647 648 649 650 651 652 653 654 655 ⬅ PontosAtendimento
31 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 309 310 311 312 313 329 332 333
4 1725 1726 1727 1728
26 516 524 525 530 531 532 533 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399
72 166 167 168 169 170 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 254 255 256 257
258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 309
310 329 330 331 332 333
55 744 745 746 747 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 1151 1152
1153 1154 1155 1156 1157 1158 1159 1177 1178 1179 1180 1201 1202 1203 1204 1217 1218 1219 1337 1435 1979 1980 1981
7 1483 1484 1485 1486 1487 1488 1489
47 1223 1224 1225 1226 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1315 1340 1344 1345 1346
1347 1348 1349 1350 1351 1400 1401 1402 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418
55 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 624 625 626 627 628 629 630 631 632
633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654
21 2074 2075 2076 2078 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2209 2280
58 393 394 395 396 414 415 416 417 418 419 420 421 422 423 424 425 427 428 2016 2017 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030
2031 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293
2294
```

Figura 2.2: Exemplo de uma instância para o problema

```
2418
0 0.0626807 0.0903784 0.183057 0.376648 0.223949 0.203054 0.339289 0.229813 0.382326 0.454119 0.465664 0.473996 0.477211 0.548027
0.557909 0.572045 0.655167 0.615543 0.498675 0.488958 0.482806 0.467985 0.468145 0.451676 0.302554 0.408416 0.399285 0.268256 0.400873
0.452755 0.471155 0.551991 0.646263 0.634093 0.465937 0.6431 0.76432 0.884652 0.897572 0.76977 0.777458 0.835118 0.899883 1.02959
0.901609 0.923599 0.783858 0.79783 0.809654 0.818784 0.889568 0.907907 0.91782 0.925874 0.912139 0.928024 0.969392 0.990107 1.05785
1.08349 0.683137 0.722906 0.748742 0.7635 0.788983 0.855286 0.936536 0.991006 1.0297 1.01802 0.932197 0.908198 0.826533 0.820898
0.806861 0.750425 0.725119 0.687068 0.653953 0.66737 0.672066 0.675612 0.719044 0.736632 0.745228 0.75212 0.790274 0.809709 0.897123
0.860353 0.795302 0.745439 0.614648 0.604536 0.446142 0.588224 0.61417 0.859282 0.961539 1.0112 1.02132 0.979876 0.881186 0.651713
0.638822 0.837087 0.906296 1.00176 1.0278 1.03558 1.11238 1.14927 1.24457 1.18052 1.17795 1.09359 0.99688 0.941179 0.932213 0.915988
0.871597 0.860729 0.846968 0.817959 0.793998 0.781065 0.753623 0.934103 0.694278 1.10339 1.14513 1.22581 1.25191 1.26945 1.3068
1.33304 1.35852 1.3902 1.47612 1.45971 1.4102 1.37371 1.32388 1.38355 1.34441 1.31055 1.21732 1.28341 1.1944 0.957801 1.12845 1.08293
1.05586 0.87467 0.96602 0.956255 0.925344 1.25445 1.13519 0.993741 1.00543 1.33939 1.62432 1.64654 1.67007 1.78519 1.81038 1.83467
1.85162 1.87023 1.74828 1.72127 1.70277 1.6675 1.74325 1.79027 1.82173 1.86622 1.90059 1.88734 1.85796 1.8404 1.81979 1.78168 1.74549
1.6973 1.65056 1.61549 1.58332 1.56689 1.54469 1.51599 1.48824 1.46065 1.41812 1.88973 1.89 1.90884 1.91186 1.91484 1.98306 1.99959
1.99013 1.92148 1.92533 1.92901 1.9328 1.93723 1.93768 1.93338 1.92918 1.92504 1.92089 1.91668 1.91051 1.9006 1.80354 1.77291 1.74948
1.74328 1.71323 1.65602 1.61169 1.59045 1.57345 1.53889 1.49621 1.44572 1.39985 1.38948 1.36236 1.28724 1.26876 1.25279 1.23668
1.22196 1.21283 1.17146 1.24399 1.40049 1.39659 1.33908 1.30549 1.29682 1.27999 1.2401 1.23061 1.19284 1.07103 1.00788 1.65174 1.71165
1.66545 1.79944 1.84675 1.91919 1.91336 1.90998 1.9137 1.91744 1.91603 1.91239 1.94694 1.97081 1.98023 1.98179 1.98332 1.98483 1.98632
1.98752 1.98874 1.98988 1.99118 1.99267 1.99416 1.99346 1.99214 1.99084 1.98936 1.9879 1.98642 1.98491 1.98337 1.98179 2.04617 2.05963
```

Figura 2.3: Exemplo da matriz de distâncias usada

3 Metodologia

A primeira parte do projeto constitui-se na construção instâncias diferentes, de forma a podermos realizar testes comparativos entre os algoritmos desenvolvidos. Foi utilizada uma parte do mapa da cidade de Juiz de Fora para a construção de um grafo e posteriormente os dados de entrada do problema foram randomizados, variando-se o número de carros e passageiros a fim de se encontrar um *insight* sobre o comportamento dos algoritmos com o crescimento das instâncias.

Sendo o problema abordado NP-difícil, como estabelecido por Balardino and Santos [2016], não se conhece algoritmos para resolvê-lo em tempo polinomial e com solução ótima. Assim, optou-se neste trabalho, por idealizar e implementar novas heurísticas para obtenção de soluções e reduzir o tempo de processamento.

São propostas duas heurísticas construtivas ambas estratégias gulosas, a primeira a fim de gerar resultados como base de comparação para os outros algoritmos propostos, com uma ideia intuitiva para a resolução do problema. A segunda implementa uma forma de montar soluções a partir de uma ordem pré estabelecida de inserções gulosas na solução, ordem essa que tem grande impacto na solução. Portanto são implementados algoritmos para tentar encontrar a ordem ótima de inserção. As metaheurísticas escolhidas foram adaptações do *Greedy Randomized Adaptive Search Procedure* (GRASP) e do Algoritmo Genético (AG), por serem algoritmos já consolidados na literatura e apresentarem abordagens diferentes quanto a construção das soluções na parte de exploração/explotação das soluções.

Devido a aleatoriedade presente em ambos os algoritmos, foram executados diversos testes em cada uma das várias instâncias criadas e medido o tempo de processamento e o valor obtido na função objetivo.

4 Revisão Bibliográfica

Para que fosse possível ter uma melhor compreensão da classe do problema abordado foi pesquisado na literatura diversos problemas de roteamento de veículos, dos quais se destaca o problema do Caixeiro-Viajante, um problema clássico na literatura de ciência da computação e conhecido pela sua complexidade, onde foi dado o enfoque às variações deste problema que possuem similaridades com o problema de transporte compartilhado com atendimento suficientemente próximo, como por exemplo o atendimento suficientemente próximo de clientes e os problemas de coleta e entrega. Também foram pesquisadas outras formulações do problema de *Rideshring* e os algoritmos utilizados para se obter as soluções. Fora da literatura foi também apresentado alguns aplicativos presentes no mercado que oferecem serviços de transporte, a fim de se encontrar implementações aplicadas dos conceitos estudados.

4.1 Caixeiro-Viajante

O problema do caixeiro-viajante (*Traveling Salesman - TS*) pode é um dos problemas mais estudados de roteamento em grafos e tem como objetivo encontrar uma rota de um caixeiro que precisa percorrer uma série de cidades e retornar à sua origem no menor caminho possível. Este é um problema conhecido NP-Difícil que possui diversas formulações, a seguir apresentaremos alguns exemplos dessas formulações que o aproximam do problema de transporte compartilhado. Bektas [2006] faz uma revisão e apresenta algumas variações das formulações do problema de caixeiro viajante com múltiplos caixeiros (*Multiple Traveling Salesman Problem* - mTSP) e das soluções já apresentadas apara o problema.

Baranwal et al. [2016] aborda problemas de atendimento suficientemente próximo para o Problema do Caixeiro Viajante, onde para o cliente ser atendido é suficiente que o caixeiro passe em um ponto próximo a ele. Este modelo foi baseado em uma necessidade real onde uma empresa utilizaria medições á distância *wireless* de contas como energia e

água. Este trabalho ainda aborda separadamente problemas com múltiplos cacheiros e o caso onde o caixeiro não precisa retornar ao seu depósito de origem.

Ghafurian and Javadian [2011] formula o problema de forma que os caixeiros podem possuir diferentes origens e destinos finais, considerandos vários depósitos considerando restrições em que os depósitos devem possuir a mesma quantidade de carros saindo e chegando.

4.2 Transporte Compartilhado

Vários problemas de *rideshraring* foram abordados com diversas formulações e objetivos, Furuhata et al. [2013] apresenta uma revisão de alguns trabalhos na área juntamente com uma proposta de classificações destes problemas a partir de um enfoque mais empresarial, comparando as abordagem com outras presentes no mercado.

Transporte compartilhado é parte de uma classe de problemas de transporte chamada "coleta e entrega"(*pickup and deliver*) onde motoristas possuem uma capacidade de carga e devem realizar coletas e entregas durante o seu percurso. Alguns destes problemas são abordados por Savelsbergh and Sol [1995] considerando diferentes abordagens, entre elas um problema de transporte de pessoas chamado no artigo de *Dial a ride problem* - Darp.

Muitos trabalhos foram publicados considerando o transporte compartilhado de pessoas entre eles se destacam os trabalhos de Herbawi and Weber [2012] onde é implementado um algoritmo genético para um problema similar mas em que são considerados múltiplos destinos e janela de tempo. Já Song et al. [2012] aplica um algoritmo que ordena passageiros de acordo com suas proximidades aos carros para melhor alocar-los aos carros. Teodorović and Dell'Orco [2008] apresentam um algorítimo de otimização por colônia de abelhas a fim de resolver o mesmo problema.

Já Spieser et al. [2015] consideram o problema com a inserção dinâmica de passageiros, considerando uma empresa com carros dirigidos automaticamente e passageiros que requerem o serviço em diferentes tempos.

Vários algoritmos diferentes como força bruta, programação linear e algumas heurísticas como busca tabu são citados por Xia et al. [2015] para a resolução do problema

de múltiplos passageiros com múltiplos destinos.

O problema de roteamento de ônibus escolar, possuí algumas variações semelhantes ao problema apresentado, neste problema deve-se determinar a rota de uma frota de ônibus para atender os alunos de uma ou mais escolas, Park and Kim [2010] faz uma revisão sobre este problema.

4.2.1 Atendimento suficientemente próximo

O problema de *ridesharing* com atendimento suficientemente próximo foi proposto por Balardino and Santos [2016], que apresentam um algoritmo construtivo a fim de gerar soluções iniciais para o problema, com uma busca local iterada (*ILS*), e um algoritmo baseado em geração de colunas para o refinamento da solução.

4.3 Aplicativos

Não foi encontrado nenhum aplicativo que incluísse o atendimento suficientemente próximo de passageiros, mostrando que esta é uma área que ainda pode ser explorada.

Existem, porém, aplicativos como o *blablacar* (https://www.blablacar.com.br/), que permite dividir os custos de viagens mais longas através da oferta de carona no aplicativo. *Hitcharide* (http://hitcharide.me/) um aplicativo que funciona como uma rede social, onde são informadas as origens, destinos e preços de viagens e seus amigos poderão visualizar e entrar em contato. *UberPool* (https://www.uber.com/pt-BR/ride/uberpool/) permite realizar o transporte urbano considerando vários passageiros que dividem as despesas da viajem.

5 Abordagens Propostas

5.1 Construtivos

Heurísticas construtivas em inteligência computacional são algoritmos que permitem obter soluções viáveis para os problemas utilizando abordagens intuitivas . São apresentados a seguir dois algoritmos construtivos, o primeiro pretende fornecer uma solução *naive*, a fim de comparações com os demais algoritmos propostos, o segundo é utilizado em conjunto com os algoritmos Genético e GRASP para a geração de boas soluções para o problema.

5.1.1 Construtivo Guloso

Este algoritmo guloso foi proposto por Balardino and Santos [2016] como construtivo inicial para seus algoritmos, e é usado neste trabalho como solução de base, a fim de comparações com os algoritmos propostos.

É considerado, para cada passo do algoritmo, a inserção de um passageiro de forma a maximizar a função objetivo. Então, é calculado o impacto causado na função objetivo para cada combinação de ponto de atendimento e ordem relativa aos passageiros já atendidos pelo carro em todas as combinações carro-passageiro que não ferem as restrições do problema. Uma vez incluída na solução, uma associação carro-passageiro não será mais removida. Este processo é descrito no Algoritmo 1.

Algoritmo 1: Pseudo-código do algoritmo construtivo guloso

```
1  início
2      enquanto ∃ Passageiro possível de inserir na solução faça
3          melhorImpacto ← 0;
4          melhorPassageiro ← 0;
5          para cada Passageiro ∉ Solução faça
6              impacto ← CalculaMelhorImpactodeInserção(Passageiro,Solução);
7              se melhorImpacto < impacto então
8                  melhorImpacto ← impacto;
9                  melhorPassageiro ← Passageiro;
10             fim se
11         fim para
12         InserePassageiro(Passageiro, melhorImpacto);
13         AtualizaSolução();
14     fim enquanto
15 fim
16 retorna Solução
```

O algoritmo para se encontrar o melhor impacto dentro da solução para um passageiro envolve, para cada ponto do passageiro, calcular o desvio ao se alocar o passageiro em cada posição disponível no carro, entre origem do motorista e ponto de atendimento do primeiro passageiro, ponto de atendimentos adjacentes entre passageiros e ponto de atendimento do último passageiro e destino, como representado pelas setas contínuas na Figura 5.1.

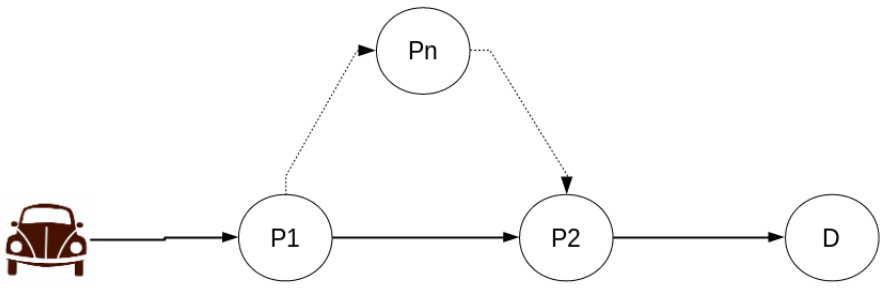

Figura 5.1: Exemplo de teste para inserção de um passageiro na rota de um carro

Então é retornado, se possível, a melhor inserção em um carro disponível, caso não sejam encontrado carros com disponibilidade de desvio e vagas para este passageiro,

o mesmo não será incluído na solução.

5.1.2 Construtivo por Ordem de Inserção

O algoritmo construtivo por ordem de inserção utiliza um conceito similar ao construtivo guloso. Porém é recebido como entrada uma ordem pré definida de inserção dos passageiros na solução, de maneira que a cada iteração, se possível, um passageiro é adicionado à solução utilizando-se a abordagem gulosa de forma que este seja alocado ao um carro para se gerar o maior impacto na função objetivo. O Algoritmo 2 apresenta a forma como é montada a solução.

Algoritmo 2: Pseudo-código do algoritmo construtivo por vetor de prioridades

```
1 início
2     para cada Passageiro ∈ VetorOrdenado faça
3         melhorImpacto ← CalculaMelhorImpactodeInserção(Passageiro,Solução);
4         InserePassageiro(Passageiro, melhorImpacto);
5         AtualizaSolução();
6     fim para
7 fim
8 retorna Solucão
```

Para este algoritmo é extremamente importante a ordem em que são inseridos os passageiros, de forma que a mudança de um passageiro nesta ordem pode mudar toda a solução devido às restrições de desvio máximos dos carros. Seja $p = |P|!$ existem $|P|!$ permutações diferentes quanto a possibilidades de ordem de incerção diferentes, foram implementados algoritmos para se encontrar uma melhor ordem de inserção, estes algoritmos serão descritos a seguir.

5.2 Buscas Locais

Um algoritmo de busca local toma como principio uma solução para o problema e aplica pequenas mudanças a esta solução para tentar melhorar sua qualidade. Foram desenvolvidas buscas locais em relação a ordem de inserção e ao ajuste fino de problemas encontrados no algoritmo construtivo por ordenação de passageiros.

5.2.1 Busca no Vetor Ordenado

A primeira busca local aplicada visa encontrar a melhor combinação no vetor ordenado para o algoritmo construtivo, modificando a ordem de inserção dos passageiros na solução. Para criar a vizinhança são considerados os N últimos passageiros na ordem de inserção e para a geração de cada solução vizinha este passageiro é adiantado em uma posição na ordem de inserção, de maneira que são testadas todas as possíveis posições deste passageiro em relação aos demais. Esta esta vizinhança foi escolhida pois os elementos no final do vetor de inserção tem mais chance de estar causando um maior impacto na função objetivo, uma vez que são os últimos a serem inseridos na solução.

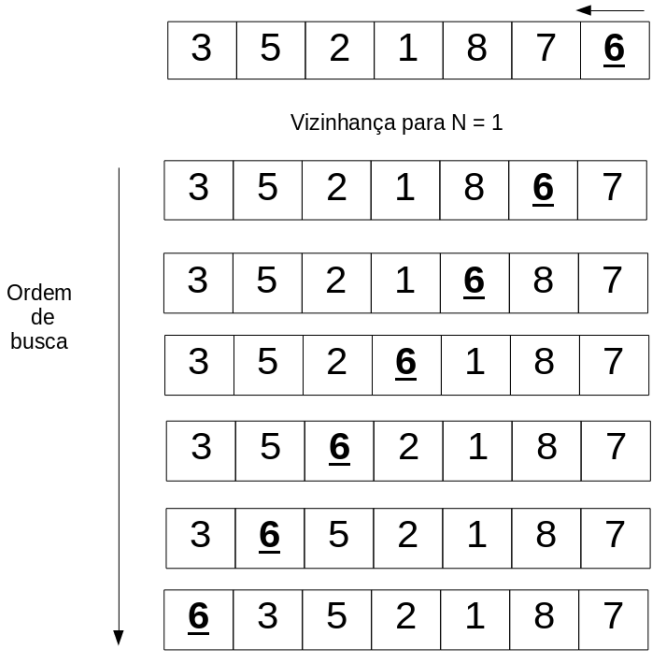

Figura 5.2: Exemplo de uma vizinhança para N = 1

Esta busca local foi implementada de maneira que se caminha no espaço de soluções ao se encontrar a primeira solução com valor de função objetivo melhor que a solução atual. Assim o processo é reiniciado a partir do novo vetor obtido. Quando, para os N últimos elementos, não é encontrada nenhuma solução com melhora no objetivo, a busca é encerrada.

5.2.2 Busca nos Pontos de atendimento

Com a inserção de passageiros de forma definitiva no algoritmo construtivo não são revisitados pontos de atendimento dos passageiros já inseridos, esta busca local tem por objetivo otimizar os pontos de atendimento de um passageiro, uma vez definida pelo construtivo a alocação e ordem de atendimento dos passageiros.

Assim para cada solução considerada, sua vizinhança é obtida modificando-se o ponto de atendimento de um passageiro, este procedimento é feito localmente para cada carro de maneira que sempre que é encontrada uma mudança no atendimento de um passageiro, a busca reinicia para o carro específico.

Algoritmo 3: Pseudo-código do algoritmo de busca nos pontos de atendimento

```
1  início
2      para cada carro k faça
3          para cada Passageiro p alocado em k faça
4              melhorPt ← TestaPontosDeAtendimento(p);
5              se melhorPt ≠ ponto atual então
6                  AtualizaSolução(p,melhorPt);
7                  Reeinicia Busca em k;
8              fim se
9          fim para
10     fim para
11 fim
12 retorna Solução
```

5.3 Grasp

GRASP, do inglês *Greedy Randomized Adaptive Search Procedure* é uma metaheurística que vêm sendo utilizada desde o final da década de 80 para problemas de otimização e visa gerar diversas soluções iniciais a partir de um algoritmo guloso randomizado e aplicar buscas locais para refinar a solução encontrada. O procedimento completo é mostrado no Algoritmo 4.

Para o este algoritmo foi usado o construtivo por Ordem de Inserção, completamente randomizado para geração de uma solução inicial, logo após é aplicada a busca com N = 2 no vetor ordenado a fim de se obter uma ordem de inserção segundo esse

movimento. Então aplica-se a busca local nos pontos dos passageiros. A condição de parada estipulada é a de 50 iterações sem melhoras na função objetivo.

Algoritmo 4: Pseudo-código do algoritmo GRASP

```
1  início
2      MelhorSolução ← GeraSolução;
3      BuscaLocalPontosPassageiro(MelhorSolução);
4      enquanto ¬Condição de parada faça
5          Solução ← GeraSolução;
6          BuscaLocalVetorOrdenado(Solução);
7          BuscaLocalPontosPassageiro(Solução);
8          se funcObjetivo(MelhorSolucão) < funcObjetivo(Solucão) então
9              MelhorSolução ← Solução;
10         fim se
11     fim enquanto
12 fim
13 retorna MelhorSolucão
```

5.4 Algoritmo genético

Algoritmos genéticos são algoritmos populacionais bioinspirados que permitem que exploremos grande parte do espaço de busca, mantendo diversas soluções diferentes e aplicando-se operações nestas soluções, procurando manter partes destas soluções que levem a bons resultados na função objetivo. Em um algoritmo genético, nos referimos ao conjunto de soluções como população onde uma solução representa um indivíduo e sua aptidão indica o quanto esta solução é interessante de se manter na população.

De forma resumida, o algoritmo genético procura manter indivíduos com boa aptidão na população, visando replicar as características desses indivíduos ao se gerar novos. Para que se obtenha este resultado são definidas operações chamadas de cruzamento, mutação e seleção, que procuram combinar as características de dois indivíduos, realizar pequenas modificações e selecionar indivíduos a serem cruzados ou removidos da população, respectivamente.

Para este problema foi desenvolvido um algoritmo genético com chaves randômicas (Bean [1994]). Este tipo de algoritmo genético monta uma solução a partir codificações de soluções a partir de vetores de valores reais, e para cada indivíduo deve-se decodificar este vetor em uma solução para o problema. O *fitness* de cada indivíduo é calculado normalmente pela função objetivo, assim podemos encontrar e classificar as soluções.

Para encontrar as soluções para este problema, no algoritmo genético testado é utilizado o algoritmo construtivo por ordem de inserção, de maneira que cada individuo da população gera uma variação da ordem de inserção na solução, montando assim uma solução diferente. Para isso é definido que cada membro da população será um vetor de chaves randômicas, batizado *Vetor de prioridades* onde cada passageiro tem um valor de prioridade associado e a ordem de inserção de um passageiro na solução é calculada ordenando-se sua prioridade.

As Tabelas 5.1 e 5.2 apresentem um exemplo de um vetor de prioridade e a ordem de inserção geradas por estas prioridades:

Passageiro	Prioridade
1	.34
2	.90
3	.12
4	.55
5	.72

Tabela 5.1: Vetor de prioridades

Ordem de inserção	Passageiro
1	2
2	5
3	4
4	1
5	3

Tabela 5.2: Ordem de inserção

Assim para a construção da população inicial foram randomizados vetores de prioridade para cada um dos indivíduos, gerados suas respectivas soluções e ordenado-se a população de acordo com sua aptidão.

5.4.1 Descrição das operações

- **Seleção**

  A seleção tem um papel importante no algoritmo genético, uma vez que a ideia por trás dos AGs é manter as características de soluções com um bom *fitness*.

  Para a seleção foi aplicado um esquema de ranking, onde cada indivíduo é ordenado e ranqueado de acordo com o seu *fitiness*, e a chance de uma posição i ser sorteada é dada pela Equação 5.1, onde a e ϵ são parâmetros passados, n é o número de indivíduos. Esta forma de seleção foi inspirada em um sorteio por fichas, onde o último colocado recebe um número a de fichas e, para cada posição acima, são acrescentadas ϵ fichas.

$$X_i = 2 * (a + n * \epsilon)/((2 * a + (n - i) * \epsilon) * n) \quad (5.1)$$

Para os experimentos apresentados nesta pesquisa são utilizados os parâmetros $a = 5$ e $\epsilon = 1$

- **Cruzamento**

  Para o cruzamento foram feitos 3 tipos de cálculos para a geração das soluções filhas.

  1. Corte simples: Se sorteia um uma posição do vetor p e o filho recebe os valores de prioridade de P1 para todos os passageiros com índice $\leq$ p, e valores de prioridade de P2 para índices $>$ p.
  2. Média: o cálculo da prioridade de um passageiro é calculada pela média simples da prioridade dos passageiro em seus pais.
  3. Média Ponderada: o cálculo da prioridade de um passageiro é calculada pela média ponderada considerando-se a aptidão de seus pais.

- **Mutação**

  A mutação foi feita sorteando um passageiro e gerando uma nova prioridade randomizada para um dos passageiros do indivíduo.

- **Gerações**

  Para cada geração são gerados e adicionados por meio de cruzamentos à solução indivíduos até que a população possua o dobro da população inicial, após estas inserções são selecionados indivíduos para remoção sendo que o indivíduo mais adaptado não participa deste sorteio, garantindo assim que a melhor solução permaneça na população. São removidos indivíduos de maneira até que sobrem 80% indivíduos da população original, finalmente gerados randomicamente os 20% restantes dos indivíduos. Este processo é ilustrado na Figura 5.3.

  Como é apresentado no Algoritmo 5 são aplicadas todas as operações descritas acima. Para os testes experimentais foram setadas probabilidades iguais para cada cruzamento descrito, 10% de probabilidade de mutação de um indivíduo. Ainda

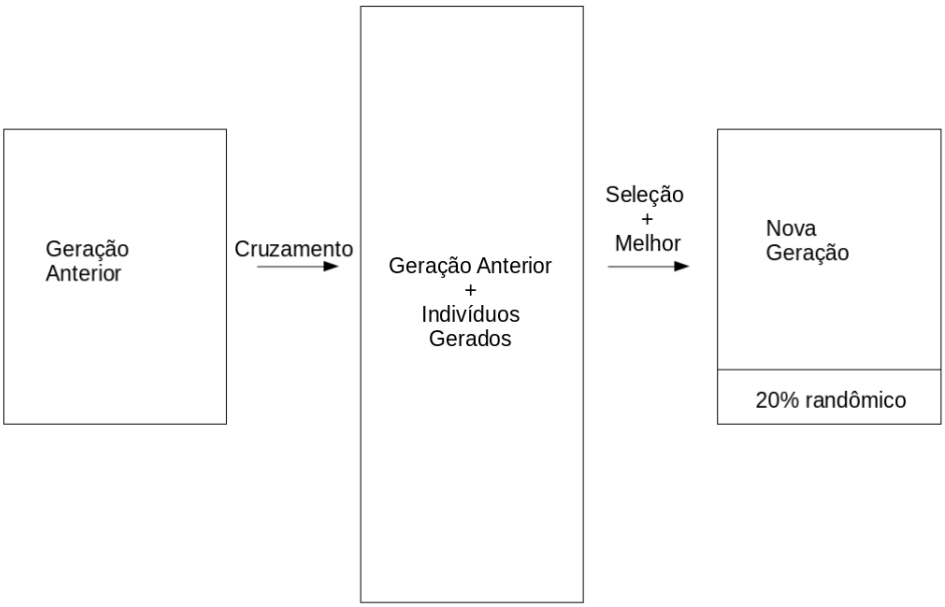

Figura 5.3: Ilustração do processo de criação de uma geração

foi testado uma variação aplicando-se a busca local nos pontos dos passageiros na melhor solução encontrada pelo genético.

Algoritmo 5: Pseudo-código do algoritmo genético com chaves randômicas

```
1  início
2      TamanhodaPopulação ← numIndivíduos;
3      InicializaPopulação(População, numIndivíduos);
4      enquanto ¬Condição de parada faça
5          enquanto TamanhodaPopulacão < 2 ∗ numIndividuos faça
6              Pais ← SorteiaIndividuosParaCruzamento(TamanhodaPopulacão);
7              filho ← Cruzamento(Pais);
8              mutação ← sorteiaMutação;
9              se mutação então
10                 Mutação(filho);
11             fim se
12             AdicionaNaPopulação(filho);
13             TamanhodaPopulacão ++;
14         fim enquanto
15         enquanto TamanhodaPopulação > .8 ∗ numIndividuos faça
16             RemoveDaPopulação(SorteiaIndividuoParaRemoção(TamanhodaPopulacão−
               1));
17         fim enquanto
18         enquanto TamanhodaPopulacão > numIndividuos faça
19             AdicionaNaPopulação(GeraIndivíduoRandômico);
20         fim enquanto
21     fim enquanto
22 fim
23 retorna MelhorSolução
```

5.5 Considerações

Estes algoritmos foram apresentados visando testar o impacto da ordem de inserção na solução por parte do algoritmo guloso por ordem de inserção, e a melhor forma de se encontrar uma boa ordem de inserção, sendo escolhido o algoritmo *GRASP* pela possibilidade de se analisar a qualidade e escalabilidade da busca no vetor proposta a partir de diversas soluções iniciais. O algoritmo genético foi proposto para se obter uma maior busca no espaço de ordens possíveis de inserção.

6 Experimentos e Resultados

6.1 Geração de Instâncias

Para a geração das instâncias de teste para os algoritmos propostos, foram retiradas as informações sobre ruas a partir de dados de um segmento de mapa real, fornecidos pelo site *Open Street Maps*(OSM) que, de acordo com Haklay and Weber [2008] é um mapa criado com licença aberta em que são mesclados dados fornecidos por instituições e edições de voluntários, a Figura 6.1 mostra um exemplo do site e como é selecionada a área desejada.

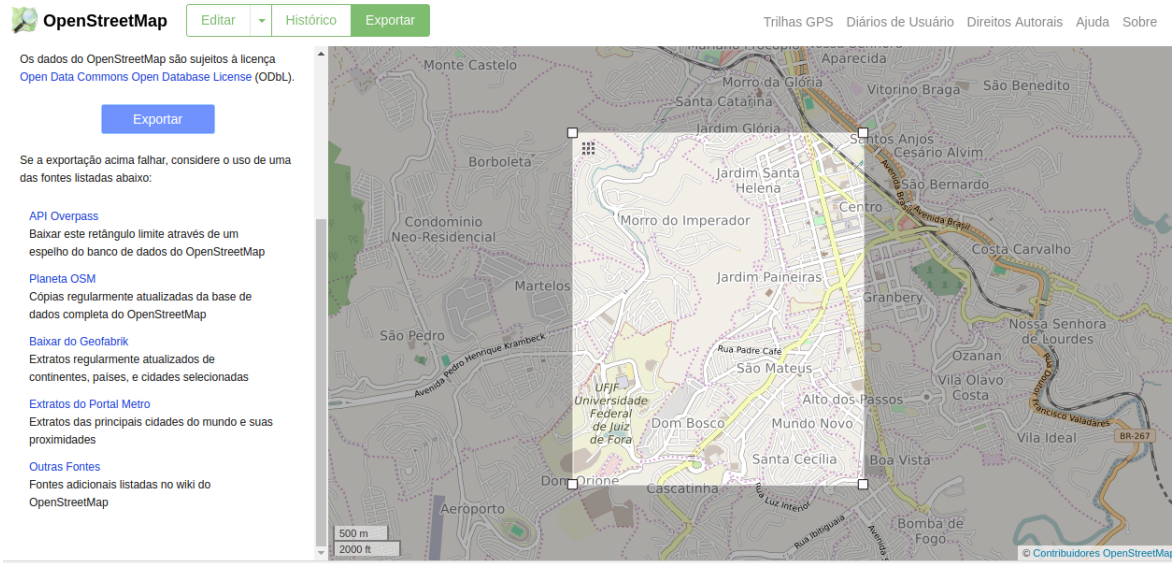

Figura 6.1: Recorte de mapa no Open Street Maps

```
<way id="23725971" visible="true" version="9" changeset="20120174" timestamp="2014-01-21T11:10:36Z" user="TrevorInserts" uid="1854688">
 <nd ref="1337049967"/>
 <nd ref="256930044"/>
 <tag k="highway" v="residential"/>
 <tag k="name" v="Rua Doutor Constantino Paleta"/>
 <tag k="oneway" v="no"/>
</way>
<way id="118573246" visible="true" version="8" changeset="29224064" timestamp="2015-03-03T14:31:54Z" user="henriqueln" uid="2637617">
 <nd ref="1333374835"/>
 <nd ref="1339776357"/>
 <nd ref="3381176226"/>
 <nd ref="3381176882"/>
 <nd ref="3381176225"/>
 <nd ref="1333374837"/>
 <tag k="highway" v="residential"/>
 <tag k="name" v="Rua Engenheiro Mauricio Giron"/>
 <tag k="oneway" v="yes"/>
```

Figura 6.2: XML com informações sobre ruas

O OSM fornece informações em um arquivo XML sobre latitude, longitude e arcos conectando pontos relevantes. Sobre as conexões, o OSM informa qual o sentido e se é

<node id="3764164422" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:38Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7826486" lon="-43.3496801"/>
<node id="3764164423" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:38Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7826670" lon="-43.3474777"/>
<node id="3764164424" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:38Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7826699" lon="-43.3486326"/>
<node id="3764164426" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:39Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7826947" lon="-43.3488071"/>
<node id="3764164427" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:39Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7827174" lon="-43.3702980"/>
<node id="3764164428" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:39Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7827179" lon="-43.3490843"/>
<node id="3764164430" visible="true" version="1" changeset="34325479" timestamp="2015-09-29T16:47:39Z" user="Vitor Dias - importação de dados" uid="1631826" lat="-21.7827726" lon="-43.3489435"/>

Figura 6.3: Exemplo de pontos do OSM

uma via de mão dupla. Para o custo das conexões, não fornecido pelo OSM, utilizou-se a distância euclidiana calculada com base nas coordenadas geográficas. O arquivo inclui pontos como árvores, rios, e conexões com pontos fora do segmento extraído. Essas informações não tem utilidade para o presente estudo e foram removidas, construindo-se dessa forma um grafo conexo e orientado, que modela as distâncias entre as pontos gerados pelo OSM, desconsiderando a altitude e curvas entre pontos. A Figura 6.4 apresenta o plotagem dos pontos (verde), e das arestas (em roxo) obtidos no final do processamento.

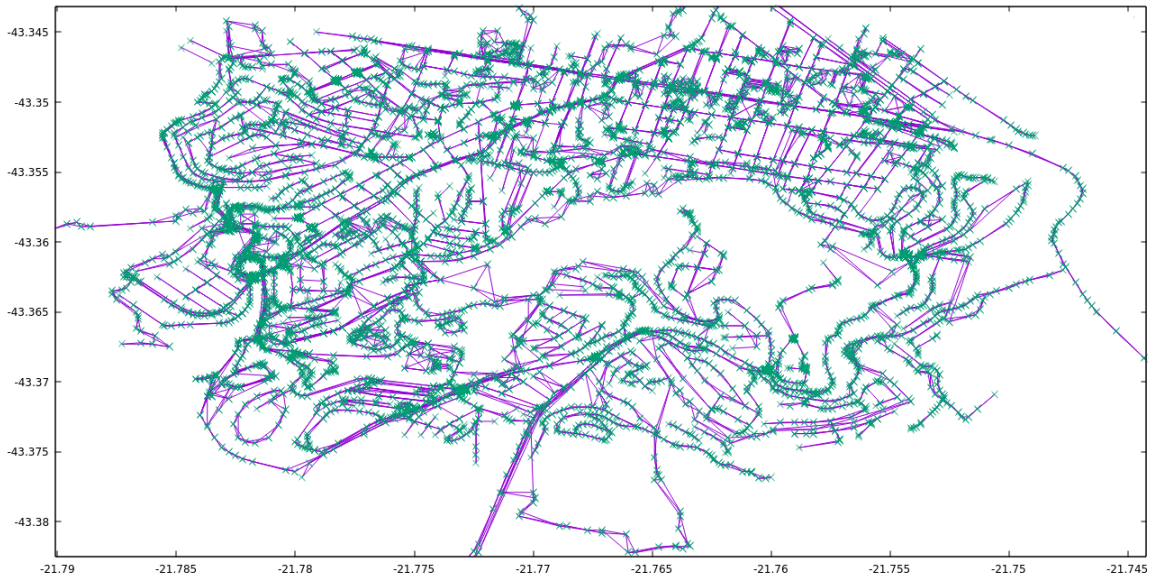

Figura 6.4: Plot do grafo gerado

Após processar o XML contendo as informações das arestas e pontos foi aplicado o algoritmo de Floyd (Floyd [1962]), que gera, a partir de um grafo, o custo do caminho mínimo entre todos os pontos, assim facilitando o processamento das informações durante a execução do programa.

O recorte do mapa da OSM utilizado corresponde à parte da região central da cidade de Juiz de Fora (MG), e gerou um grafo com 2.418 nós. Entre esses nós foram

escolhidos aleatoriamente pontos de origem para passageiros e carros. Nenhum ponto de origem a uma distância do destino menor do que o limite de deslocamento de passageiros foi admitido, dessa forma, exclui-se a possibilidade do passageiro alcançar o destino caminhando.

O diâmetro do grafo gerado é igual a 9, portanto, arbitramos 10 para o valor da variável L, garantindo a priorização do atendimento ao passageiro. Também foi arbitrado para D_{max} o valor 1,5 garantindo assim um desvio máximo igual a 50% da rota mínima dos motoristas.

Para os testes foram criadas instâncias com 20, 30, 40 e 50 carros e 25, 50, 75, 100, 125, 150, 175 e 200 passageiros, sendo que para cada combinação carro-passageiro, 5 instâncias diferentes de teste foram geradas totalizando 160 entradas diferentes.

6.2 Experimentos computacionais

Foram realizados 10 execuções para cada uma das 160 instâncias geradas, onde foi medido o tempo de execução, o resultado encontrado na função objetivo e o número de passageiros atendidos. Os gráficos nas Imagens 6.5 e 6.6 apresentam a média dos resultados e tempos de execução obtidos ao nas execuções das 5 instâncias que representam cada uma das combinações carro-passageiro.

Todos os testes foram feitos em uma máquina Dell i15-5557-A15, com sistema Ubuntu 16.04, onde o código foi implementedo na linguagem C++, e estará disponível em http://www.monografias.ice.ufjf.br.

Para o AG a condição de parada foi estipulada para estes testes como 20 gerações sem uma melhora na função objetivo, enquanto para o *Grasp* 50 iterações sem apresentar melhora. A estipulação destes valores foi arbitrária, uma vez que não se pode prever quanto tempo será gasto ou quantas soluções serão geradas pela busca local apresentada.

Como é possível se observar nos gráficos da Figura 6.5 o algoritmo genético combinado com a busca local apresentou resultados ligeiramente melhores que os outros algoritmos apresentados, e isso é verificado na tabela 6.1, que mostra a obtenção da melhor média em 139 (85, 8%) instâncias, melhor solução individual em 126 (76, 4%), e melhor resultado para o intervalo de confiança calculado em 66 instâncias. Os resultados são

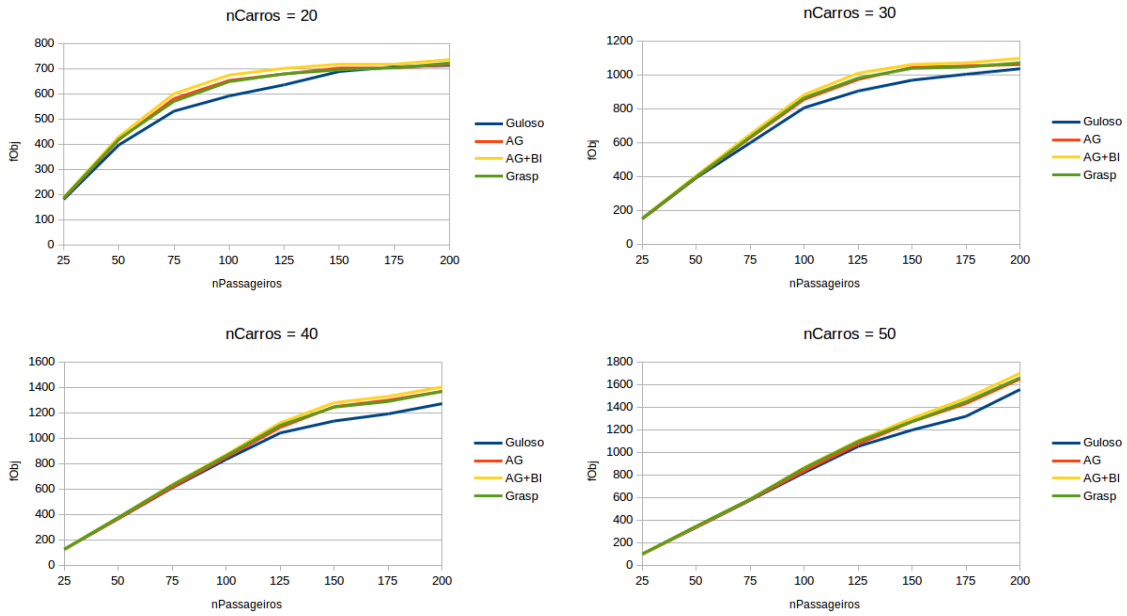

Figura 6.5: Gráficos com médias dos valores de função objetivo encontrados para cada conjunto de instâncias

apresentados nas Tabelas A.1, A.2, A.3 e A.4 mostrando os valores médios obtidos para cada instância e o intervalo com confiança de 98%, obtido a partir de um teste *t-student*.

Pode-se observar também, como esperado, que os valores obtidos aumentam linearmente com o aumento do número de passageiros enquanto este não ultrapassar o número de vagas totais da entrada, e um aumento aparentemente logarítmico, tendendo ao valor da função objetivo com desvio igual a 0, para quantidades de passageiro maiores que as vagas disponíveis.

Tabela 6.1: Avaliação por instância dos resultados

Algoritmos	Número de Instâncias		
	Melhor resultado	Melhor média	IC = 98%
Guloso	2	4	3
AG	2	1	1
AG+BL	129	139	66
Grasp	32	18	2
TOTAL	165	162	72

As melhores soluções encontradas para cada instância são mostradas no Apêndice B.

A Figura 6.6 apresenta a média do tempo de execução para cada conjunto de instâncias, nestes gráficos podemos ver que os algoritmos genéticos apresentam um maior tempo de execução para as entradas dadas, porém estes apresentam um crescimento

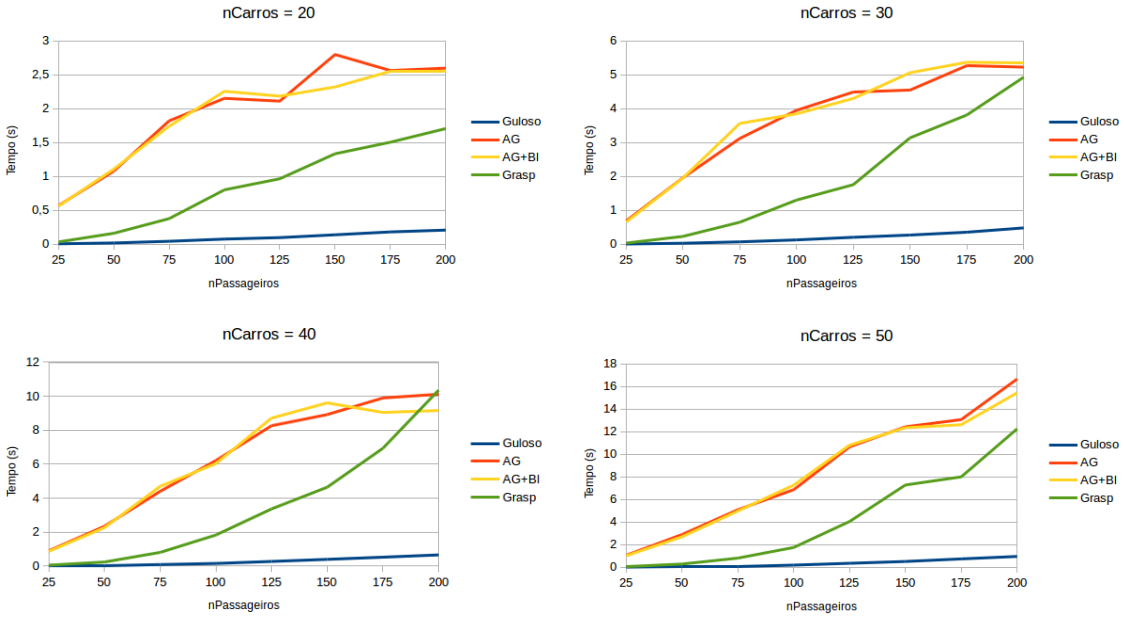

Figura 6.6: Média dos tempos de execução para cada conjunto de instâncias

linear no tempo de execução com o crescimento do número de passageiros nas instâncias, foi observado também que a aplicação da busca local nos pontos dos passageiros não apresentou um aumento no tempo de execução relevante, enquanto seu impacto na função objetivo é considerável.É possível verificar também um maior crescimento em relação ao número de passageiros nos tempos apresentados pelo algoritmo GRASP.

7 Conclusões

Este trabalho propôs duas heurísticas para a construção de soluções para o problema de transporte compartilhado com atendimento suficientemente próximo. Os experimentos realizados foram realizados a partir de dados retirados de uma cidade real e com condições iniciais para o problema randomizadas. Foi verificado experimentalmente o melhor desempenho apresentado pelo AG implementado com uma busca local aplicada no final da execução.

Também foi verificada a eficiência da nova abordagem proposta para o problema, em se considerar a construção de soluções a partir da atribuição de passageiros aos carros em relação a abordagem tradicional de considerar os carros atendendo passageiros.

Trabalhos futuros podem incluir comparações com algoritmos baseados em construtivos diferentes, uma abordagem híbrida considerando o algoritmo Guloso para prover possíveis indivíduos para o AG pode ser explorada uma vez que este apresentou bons resultados com um baixo tempo de execução. Outra proposta é analisar uma formulação mais geral do problema, com diversos destinos, podendo ou não ser aplicado o atendimento suficientemente próximo aos destinos.

Referências Bibliográficas

Allan F Balardino and André G Santos. Heuristic and exact approach for the close enough ridematching problem. pages 281–293, 2016.

Allan Fernandes Balardino et al. Transporte compartilhado com atendimento suficientemente próximo de passageiros. 2016.

Mayank Baranwal, Brian Roehl, and Srinivasa M Salapaka. A deterministic annealing approach to the multiple traveling salesmen and related problems. *arXiv preprint arXiv:1604.04169*, 2016.

James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160, 1994.

Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46, 2013.

Soheil Ghafurian and Nikbakhsh Javadian. An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. *Applied Soft Computing*, 11(1):1256–1262, 2011.

Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

Wesam Mohamed Herbawi and Michael Weber. A genetic and insertion heuristic algorithm for solving the dynamic ridematching problem with time windows. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 385–392. ACM, 2012.

Nelson Avella Netto and Heidy Rodriguez Ramos. Estudo da mobilidade urbana no contexto brasileiro. *Revista de Gestão Ambiental e Sustentabilidade-GeAS*, 6(2):59–72, 2017.

Junhyuk Park and Byung-In Kim. The school bus routing problem: A review. *European Journal of operational research*, 202(2):311–319, 2010.

Martin WP Savelsbergh and Marc Sol. The general pickup and delivery problem. *Transportation science*, 29(1):17–29, 1995.

Chaochao Song, Hongguo Wang, Yanhui Ding, and Fuping Yang. Heuristic algorithm for multi-vehicle ride matching problem. In *Information Technology in Medicine and Education (ITME), 2012 International Symposium on*, volume 2, pages 1036–1040. IEEE, 2012.

Kevin Spieser, Samitha Samaranayake, Wolfgang Gruel, and Emilio Frazzoli. Shared-vehicle mobility-on-demand systems: A fleet operator's guide to rebalancing empty vehicles. 2015.

Dušan Teodorović and Mauro Dell'Orco. Mitigating traffic congestion: solving the ride-matching problem by bee colony optimization. *Transportation Planning and Technology*, 31(2):135–152, 2008.

Jizhe Xia, Kevin M Curtin, Weihong Li, and Yonglong Zhao. A new model for a carpool matching service. *PloS one*, 10(6):e0129257, 2015.

A Resultados por instância

Tabela A.1: Resultados para instancias com 20 carros

nPas	Instância	Guloso		Genetico		Genetico+BL		Grasp	
		Z	Tempo	Z	Tempo	Z	Tempo	Z	Tempo
25	1	174.007 ± 1.3612e-05	0.0074662 ± 155.221	174.458 ± 0	0.694717 ± 0.0255994	176.167 ± 1.93858	0.70399 ± 0.0974482	**177.959 ± 2.6658**	0.0404046 ± 0.0129759
25	2	185.162 ± 0	0.0046644 ± 165.175	185.227 ± 1.3612e-05	0.491423 ± 0.0304844	**186.539 ± 0.763543**	0.549096 ± 0.134813	185.704±0.965571	0.0305774±0.0114201
25	3	170.514 ± 1.3612e-05	0.0048141 ± 152.108	187.921 ± 0.690116	0.46703 ± 0.0771203	**193.877 ± 2.85569**	0.458839 ± 0.0643623	190.397±5.16729	0.0264463±0.0106918
25	4	190.377 ± 1.3612e-05	0.0055825 ± 169.826	191.848 ± 0	0.521814 ± 0.0268188	**193.799 ± 1.16109**	0.48423 ± 0.0088496	193.2±1.04279	0.0351319±0.0115747
25	5	181.324 ± 1.3612e-05	0.0057419 ± 161.75	181.451 ± 1.3612e-05	0.672266 ± 0.0338647	**186.729 ± 4.23353**	0.612706 ± 0.0812513	184.92±3.64834	0.0349145±0.00913762
50	1	420.336 ± 0	0.0220302 ± 374.953	418.61 ± 0.839604	1.06768 ± 0.0600492	**428.142 ± 4.68242**	1.02497 ± 0.0855434	426.403±5.21247	0.133279±0.0426878
50	2	385.757 ± 2.72241e-05	0.0169781 ± 344.11	410.389 ± 1.02082	1.26897 ± 0.248779	**423.891 ± 4.75171**	1.1416 ± 0.242126	408.023±4.99955	0.191834±0.0866066
50	3	389.166 ± 0	0.0170451 ± 347.151	415.918 ± 1.27703	0.911768 ± 0.0608072	**433.188 ± 8.96437**	1.06337 ± 0.272331	420.599±8.64235	0.145652±0.0637244
50	4	399.709 ± 0	0.0197137 ± 356.554	418.704 ± 1.23097	1.20914 ± 0.095058	**433.885 ± 3.9966**	1.45684 ± 0.472224	419.638±8.25652	0.162369±0.0607232
50	5	387.507 ± 2.72241e-05	0.0143035 ± 345.674	419.86 ± 0.554944	0.93697 ± 0.12177	**431.575 ± 4.3368**	0.850815 ± 0.103833	424.54±5.52506	0.171567±0.0312653
75	1	544.786 ± 5.44482e-05	0.0475107 ± 485.95	591.047 ± 3.84351	1.64696 ± 0.109972	**605.238 ± 7.60279**	1.54721 ± 0.119312	577.897±14.3168	0.389637±0.0727565
75	2	473.341 ± 0	0.0408353 ± 422.221	505.061 ± 3.20379	1.7653 ± 0.344818	**526.089 ± 8.71047**	1.62317 ± 0.323889	487.292±7.6007	0.357929±0.173276
75	3	584.79 ± 0	0.0484434 ± 521.635	644.216 ± 1.74482	2.05136 ± 0.424497	**674.127 ± 4.72675**	2.24697 ± 0.614309	646.624±9.21409	0.369713±0.133623
75	4	502.317 ± 2.72241e-05	0.0423029 ± 448.068	556.805 ± 4.56841	1.52926 ± 0.1867	**584.633 ± 6.698**	1.57269 ± 0.318328	550.263±12.9857	0.389768±0.212123
75	5	548.812 ± 0	0.0359746 ± 489.551	605.691 ± 7.16229	2.10614 ± 0.690159	**608.814 ± 4.87106**	1.70996 ± 0.289505	585.277±8.79065	0.376721±0.140028
100	1	453.434 ± 2.72241e-05	0.062772 ± 404.443	544.318 ± 5.7253	1.92265 ± 0.11169	**561.907 ± 10.0517**	2.08586 ± 0.410749	528.541±19.1398	1.03661±0.480559
100	2	585.842 ± 5.44482e-05	0.0760418 ± 522.549	663.834 ± 3.44599	1.85534 ± 0.203522	**687.126 ± 8.01874**	1.87336 ± 0.197748	650.343±15.4249	0.803199±0.32549
100	3	643.931 ± 5.44482e-05	0.0794897 ± 574.366	715.226 ± 1.27453	2.20824 ± 0.123481	**737.712 ± 6.51197**	2.91753 ± 0.558821	711.834±10.1406	0.6401±0.223752
100	4	690.997 ± 0	0.0851943 ± 616.348	702.296 ± 1.25525	2.76992 ± 0.724528	**721.319 ± 8.4346**	2.31365 ± 0.352679	705.514±8.49301	0.675066±0.310481
100	5	581.265 ± 5.44482e-05	0.0742074 ± 518.468	637.989 ± 4.7064	2.00059 ± 0.159069	**661.643 ± 5.55274**	2.08472 ± 0.337426	642.003±8.43024	0.851363±0.453671
125	1	704.831 ± 0	0.112769 ± 628.664	720.27 ± 2.9934	1.93587 ± 0.117203	**746.656 ± 7.70033**	2.20581 ± 0.392213	737.838±7.3419	0.995269±0.335866
125	2	632.761 ± 5.44482e-05	0.101401 ± 564.382	674.124 ± 3.09322	1.73298 ± 0.173143	**692.585 ± 5.96367**	2.30697 ± 0.87118	667.66±8.62993	0.845533±0.342027
125	3	585.098 ± 5.44482e-05	0.090539 ± 521.873	659.627 ± 3.28429	2.65762 ± 0.553232	**682.573 ± 4.94768**	2.23815 ± 0.323829	653.652±14.1204	1.1046±0.349554
125	4	659.308 ± 5.44482e-05	0.0985011 ± 588.067	652.282 ± 1.93139	1.94006 ± 0.324248	**669.615 ± 4.66327**	1.82736 ± 0.0840099	654.592±9.39217	0.956406±0.349203
125	5	593.94 ± 0	0.0816134 ± 529.768	684.848 ± 4.47248	2.28644 ± 0.561702	**711.991 ± 9.15558**	2.34087 ± 0.642526	680.487±13.7408	0.921644±0.343957
150	1	**742.11 ± 5.44482e-05**	0.132561 ± 661.902	723.183 ± 1.48972	2.81481 ± 0.331102	733.866 ± 6.1392	2.04075 ± 0.338154	733.713±7.49729	1.23766±0.463545
150	2	692.759 ± 5.44482e-05	0.148811 ± 617.863	707.268 ± 2.40203	3.24199 ± 1.07406	**722.431 ± 4.52943**	2.70232 ± 0.700817	700.608±6.1261	1.06521±0.203381
150	3	682.911 ± 5.44482e-05	0.140697 ± 609.085	701.385 ± 3.0283	2.54007 ± 0.456744	**721.041 ± 4.84435**	2.39813 ± 0.135776	690.419±7.32875	1.25543±0.532521
150	4	668.87 ± 5.44482e-05	0.164058 ± 596.538	688.837 ± 9.21062	2.61866 ± 0.438674	**706.21 ± 7.42801**	2.3312 ± 0.275858	674.671±13.2482	1.34788±0.430349
150	5	654.01 ± 5.44482e-05	0.105657 ± 583.334	687.064 ± 3.61466	2.77253 ± 0.552119	**695.408 ± 3.3129**	2.12373 ± 0.305777	674.23±6.85868	1.76021±0.95855
175	1	**741.706 ± 0**	0.179823 ± 661.5	717.165 ± 1.82395	2.26511 ± 0.0858085	722.089 ± 3.05394	2.18212 ± 0.591937	713.236±4.09837	1.15521±0.327137
175	2	696.532 ± 5.44482e-05	0.171035 ± 621.209	705.821 ± 2.91693	2.75272 ± 0.704946	**730.926 ± 5.6333**	2.29388 ± 0.541022	704.944±11.0788	1.49613±0.705518
175	3	667.061 ± 0	0.185249 ± 594.905	688.403 ± 2.79312	2.75448 ± 0.479952	**714.232 ± 5.64006**	3.00125 ± 0.888254	701.55±14.4281	1.6092±0.705656
175	4	**726.92 ± 0**	0.201345 ± 648.29	708.317 ± 1.74398	2.738 ± 0.142303	712.433 ± 1.76495	2.63224 ± 0.351604	702.18±4.27992	1.12921±0.405372
175	5	700.422 ± 5.44482e-05	0.165506 ± 624.684	701.014 ± 2.00274	2.29668 ± 0.304411	**706.27 ± 3.20933**	2.67153 ± 0.791061	690.989±6.61402	2.13139±1.03179
200	1	694.054 ± 5.44482e-05	0.200768 ± 618.972	719.318 ± 1.45699	2.80008 ± 0.583304	**742.177 ± 6.48214**	2.6614 ± 0.488896	736.607±9.56346	1.8545±0.531485
200	2	722.749 ± 0	0.212163 ± 644.56	712.607 ± 1.52259	2.71241 ± 0.373662	**738.245 ± 10.8642**	3.20251 ± 0.532306	730.009±12.2082	1.67801±0.625993
200	3	702.315 ± 0	0.223563 ± 626.321	718.742 ± 1.42338	2.44771 ± 0.172337	**739.763 ± 4.34657**	2.38653 ± 0.327272	708.883±11.2742	1.51791±0.491557
200	4	704.03 ± 5.44482e-05	0.207642 ± 627.865	714.964 ± 2.10408	3.02062 ± 1.09998	**722.399 ± 2.91471**	2.4952 ± 0.346314	706.701±11.8069	1.8918±0.914035
200	5	**738.761 ± 5.44482e-05**	0.192721 ± 658.861	715.128 ± 2.3662	2.00204 ± 0.202076	734.303 ± 5.7957	2.00276 ± 0.252788	727.529±12.2004	1.58449±0.490611

Tabela A.2: Resultados para instancias com 30 carros

nPas	Instância	Guloso		Genetico		Genetico+BL		Grasp	
		Z	Tempo	Z	Tempo	Z	Tempo	Z	Tempo
25	1	145.433 ± 0	0.0051837 ± 129.733	145.433 ± 0	0.641822 ± 0.0196723	**147.41 ± 1.52809**	0.658213 ± 0.0957441	146.056±1.26229	0.0308975±0.00762383
25	2	147.526 ± 0	0.0063333 ± 131.599	150.216 ± 1.3612e-05	0.653072 ± 0.0171176	153.037 ± 1.72502	0.654723 ± 0.033966	**153.612 ± 2.78106**	0.0418378 ± 0.0177385
25	3	138.801 ± 1.3612e-05	0.0064538 ± 123.816	139.395 ± 0	0.660099 ± 0.0580538	**141.515 ± 0.945576**	0.675476 ± 0.128409	140.9±0.946772	0.0332845±0.00780132
25	4	153.803 ± 0	0.0041233 ± 137.201	153.907 ± 0	0.564288 ± 0.0431792	**162.543 ± 2.63509**	0.559692 ± 0.074481	158.501±3.28582	0.0284402±0.00825424
25	5	160.844 ± 0	0.005823 ± 143.48	161.203 ± 0	0.917484 ± 0.167003	**166.014 ± 1.2818**	0.718575 ± 0.103219	164.366±2.84867	0.0329721±0.00785237
50	1	393.761 ± 0	0.0302032 ± 351.239	394.061 ± 0.0337341	2.30118 ± 0.441353	**404.422 ± 4.15868**	2.17922 ± 0.540025	403.977±2.66031	0.23943±0.0645101
50	2	386.093 ± 0	0.0318948 ± 344.397	391.509 ± 1.26996	1.65852 ± 0.216223	**404.9 ± 2.60978**	1.63702 ± 0.159911	393.562±3.35802	0.201931±0.0563051
50	3	400.193 ± 2.72241e-05	0.0215739 ± 356.984	403.746 ± 0.182282	1.78815 ± 0.279821	**410.165 ± 2.06457**	1.59922 ± 0.171605	405.51±4.81421	0.195924±0.0638606
50	4	373.352 ± 2.72241e-05	0.0301979 ± 333.032	375.71 ± 0.441337	1.90285 ± 0.153231	**379.814 ± 1.89886**	2.08523 ± 0.440698	371.521±4.89043	0.271839±0.137977
50	5	405.943 ± 2.72241e-05	0.0299215 ± 362.106	408.817 ± 0.210858	2.14891 ± 0.175737	**422.665 ± 4.26047**	2.26129 ± 0.314181	413.714±5.77701	0.242582±0.103252
75	1	561.119 ± 5.44482e-05	0.0598614 ± 500.509	616.323 ± 2.84793	2.57261 ± 0.164147	**648.776 ± 9.40499**	2.58303 ± 0.303552	626.629±10.094	0.389752±0.109314
75	2	629.217 ± 5.44482e-05	0.0814116 ± 561.238	632.468 ± 0.171856	3.78692 ± 0.468711	645.685 ± 3.00512	5.44795 ± 1.5476	**645.699 ± 3.47964**	0.707674 ± 0.259228
75	3	633.189 ± 0	0.068658 ± 564.793	632.048 ± 1.83785	2.8802 ± 0.545226	**661.332 ± 9.25056**	3.59239 ± 0.840041	642.091±9.67951	0.762582±0.402129
75	4	562.952 ± 5.44482e-05	0.0612891 ± 502.143	628.259 ± 3.59983	2.54491 ± 0.130053	**632.93 ± 3.67492**	2.60471 ± 0.180876	611.65±10.4566	0.530898±0.235829
75	5	604.833 ± 0	0.0758032 ± 539.491	629.435 ± 2.13184	3.81643 ± 0.24214	**658.408 ± 4.76109**	3.59019 ± 0.318236	637.206±5.81435	0.850321±0.2422
100	1	884.372 ± 5.44482e-05	0.13768 ± 788.806	886.931 ± 0.726874	4.98024 ± 0.754868	**912.586 ± 7.81076**	4.51297 ± 0.780417	911.483±9.63315	1.72492±1.09214
100	2	801.003 ± 0	0.125439 ± 714.446	860.571 ± 2.77311	5.14428 ± 1.43375	**897.257 ± 5.21005**	4.81978 ± 1.32745	881.288±9.94253	1.60664±0.816746
100	3	786.836 ± 0	0.133127 ± 701.801	835.884 ± 2.10256	3.0426 ± 0.105869	**842.331 ± 4.4552**	3.21982 ± 0.390098	820.407±14.2633	1.11732±0.631241
100	4	793.532 ± 5.44482e-05	0.11925 ± 707.786	851.25 ± 3.65067	3.46116 ± 0.264252	**873.86 ± 6.35115**	3.63099 ± 1.02687	846.612±9.74228	1.05066±0.368981
100	5	760.754 ± 5.44482e-05	0.124183 ± 678.542	838.645 ± 4.85193	3.12255 ± 0.163415	**887.412 ± 14.0117**	3.0527 ± 0.308222	860.722±15.1883	1.0088±0.478341
125	1	917.742 ± 5.44482e-05	0.155995 ± 818.559	987.555 ± 5.06284	3.78012 ± 0.307439	**1012.83 ± 7.88934**	3.81595 ± 0.550047	988.613±15.0506	1.63595±0.720799
125	2	858.423 ± 0	0.191517 ± 765.61	947.324 ± 3.37053	4.56366 ± 0.772953	**980.048 ± 9.02652**	4.00298 ± 0.287138	949.725±13.8969	1.91304±0.710328
125	3	823.34 ± 5.44482e-05	0.163707 ± 734.338	916.228 ± 3.73933	3.76219 ± 0.546278	**969.891 ± 7.07237**	4.30772 ± 1.41941	937.815±21.5082	1.13862±0.430199
125	4	949.186 ± 5.44482e-05	0.257908 ± 846.518	974.586 ± 5.12451	5.09298 ± 0.635732	**999.69 ± 8.53588**	4.31757 ± 0.47975	962.449±12.0741	1.81637±0.822628
125	5	970.7 ± 5.44482e-05	0.252344 ± 865.716	1034.72 ± 4.32693	5.24675 ± 1.1585	**1082.38 ± 10.0554**	5.0551 ± 0.616207	1061.92±15.9289	2.27731±0.640344
150	1	1027.1 ± 0.000108896	0.270506 ± 916.012	1072.7 ± 1.11885	4.07888 ± 0.131635	**1077.87 ± 2.39764**	4.40844 ± 0.438737	1064.09±3.52308	2.92075±1.27066
150	2	968.786 ± 0	0.238183 ± 864.021	1056.58 ± 2.45324	4.71043 ± 0.507922	**1068.07 ± 3.59872**	5.56374 ± 1.62967	1036.16±20.909	2.97138±1.22945
150	3	967.089 ± 0	0.281259 ± 862.468	1048.88 ± 4.90247	4.70719 ± 0.209341	**1066.08 ± 8.57715**	5.60199 ± 1.71011	1052.1±19.1241	3.33886±1.57258
150	4	996.49 ± 0.000108896	0.326957 ± 888.656	1051.77 ± 1.99678	4.78948 ± 0.80575	**1087.13 ± 10.0825**	5.05174 ± 0.954512	1069.59±16.6649	3.38764±1.49096
150	5	879.716 ± 5.44482e-05	0.232987 ± 784.568	986.043 ± 6.70722	4.45586 ± 0.854583	**1004.32 ± 10.6832**	4.67321 ± 1.74243	966.749±16.1564	3.07874±1.59325
175	1	1024.84 ± 0	0.330408 ± 913.943	1056.08 ± 5.2332	6.01364 ± 1.43949	**1058.17 ± 5.5914**	5.55172 ± 1.1649	1028.07±10.0377	2.84165±1.42043
175	2	1041.51 ± 0.000108896	0.422554 ± 928.732	1082.61 ± 2.80669	5.54116 ± 0.297545	**1099.89 ± 4.44795**	6.94673 ± 1.42191	1084.09±14.41	3.85335±1.31311
175	3	1041.48 ± 0.000108896	0.351195 ± 928.769	1068.09 ± 2.50657	5.48026 ± 0.92835	**1075.42 ± 4.60372**	5.65184 ± 2.14553	1049.7±12.8228	4.4074±2.70991
175	4	926.282 ± 0.000108896	0.278134 ± 826.068	991.346 ± 3.84336	3.98846 ± 0.711475	**1014.54 ± 8.35432**	3.73376 ± 0.420156	987.869±17.3416	2.83403±1.00608
175	5	980.37 ± 0	0.386886 ± 874.222	1060.32 ± 3.48692	5.32179 ± 0.565032	**1099.41 ± 7.67424**	4.95611 ± 0.716612	1074.85±20.4676	5.13246±2.69875
200	1	1079.53 ± 0.000108896	0.526066 ± 962.556	1084.06 ± 2.12522	6.06399 ± 0.243439	**1111.18 ± 7.96587**	6.24777 ± 1.17321	1088.43±9.81644	4.72504±1.7525
200	2	1058.01 ± 0.000108896	0.482728 ± 943.397	1046.7 ± 4.62188	4.5024 ± 0.2187	**1087.95 ± 13.6293**	3.96987 ± 0.230758	1064.53±10.4899	4.71634±2.01285
200	3	1054.3 ± 0.000108896	0.483407 ± 940.087	1072.07 ± 1.60861	5.21428 ± 0.49565	**1105.53 ± 8.32269**	6.09058 ± 1.33068	1092.57±7.32791	4.04347±1.15201
200	4	1026.62 ± 0	0.460386 ± 915.415	1052.87 ± 3.18083	5.24402 ± 0.478944	**1097.19 ± 11.1276**	5.37601 ± 1.21709	1061.09±22.556	4.74842±1.49047
200	5	957.492 ± 5.44482e-05	0.453544 ± 853.753	1044.46 ± 4.32532	5.10109 ± 0.425185	**1080.91 ± 11.2075**	5.06739 ± 0.421033	1045.58±12.9897	6.40386±2.23919

Tabela A.3: Resultados para instancias com 40 carros

nPas	Instância	Guloso		Genetico		Genetico+BL		Grasp	
		Z	Tempo	Z	Tempo	Z	Tempo	Z	Tempo
25	1	115.767 ± 1.3612e-05	0.0065164 ± 103.267	115.941 ± 0	0.753795 ± 0.0566242	**116.301 ± 0.425628**	0.724986 ± 0.105116	115.179±4.98062	0.0530355±0.0141187
25	2	126.991 ± 1.3612e-05	0.0063095 ± 113.28	127.142 ± 0.000267789	0.799152 ± 0.0305521	**129.551 ± 2.41202**	0.835708 ± 0.165882	128.084±1.45043	0.0427316±0.0158721
25	3	119.576 ± 6.80602e-06	0.0052676 ± 106.666	119.631 ± 0	0.704761 ± 0.0188747	125.193 ± 3.1509	0.676748 ± 0.0333877	**125.834 ± 1.96466**	0.0549895 ± 0.0263507
25	4	132.543 ± 1.3612e-05	0.0097685 ± 118.23	134.043 ± 1.3612e-05	0.959279 ± 0.180097	**134.219 ± 0.223113**	0.945917 ± 0.23312	132.401±0.326845	0.047994±0.0196064
25	5	126.882 ± 0	0.0108648 ± 113.179	127.073 ± 0	1.2851 ± 0.27229	127.122 ± 0.0703684	1.14749 ± 0.132055	**127.213 ± 0.181909**	0.0611678 ± 0.0185907
50	1	360.206 ± 2.72241e-05	0.0393972 ± 321.297	362.901 ± 0.0135557	2.54894 ± 0.559945	**375.139 ± 7.1028**	2.28798 ± 0.0962452	372.545±5.83584	0.299359±0.121546
50	2	372.272 ± 0	0.0368708 ± 332.063	372.272 ± 0	2.15646 ± 0.140699	379.986 ± 4.99411	2.12396 ± 0.219942	**381.471 ± 5.54164**	0.197038 ± 0.0360088
50	3	379.431 ± 2.72241e-05	0.0367559 ± 338.449	383.723 ± 2.72241e-05	2.40613 ± 0.15696	388.338 ± 4.15965	2.48163 ± 0.354625	**389.268 ± 4.89846**	0.199698 ± 0.0237931
50	4	348.854 ± 0	0.0338771 ± 311.175	348.931 ± 2.72241e-05	2.56013 ± 0.0191631	354.255 ± 1.64231	2.43079 ± 0.0297282	**355.445 ± 3.12205**	0.248331 ± 0.0839386
50	5	367.184 ± 2.72241e-05	0.0338522 ± 327.527	367.808 ± 0.00651257	2.05921 ± 0.0574663	**378.406 ± 4.42645**	1.9862 ± 0.130188	376.479±4.29739	0.249739±0.114719
75	1	604.537 ± 5.44482e-05	0.0941316 ± 539.211	604.862 ± 0.103757	4.37641 ± 0.426309	**619.239 ± 3.91464**	4.7336 ± 0.466655	619.13±3.5397	0.602771±0.136255
75	2	619.151 ± 5.44482e-05	0.0808368 ± 552.259	618.479 ± 0.578507	4.68942 ± 0.923661	**634.081 ± 5.98023**	4.17775 ± 0.408017	627.231±5.49366	0.991956±0.286505
75	3	600.231 ± 0	0.0825025 ± 535.38	616.446 ± 1.03069	3.34173 ± 0.160528	**640.917 ± 8.2101**	3.96539 ± 0.885106	629.939±6.2676	0.588225±0.220364
75	4	591.939 ± 0	0.0846667 ± 527.981	595.299 ± 0.204804	5.00961 ± 1.35237	**608.175 ± 5.08851**	5.14979 ± 1.40332	606.525±5.51531	0.811576±0.428168
75	5	629.294 ± 0	0.102199 ± 561.289	633.708 ± 0.798231	4.5886 ± 0.615323	**666.903 ± 6.78967**	5.43723 ± 1.16811	661.197±9.80317	1.03261±0.22556
100	1	836.777 ± 0	0.148891 ± 746.338	855.841 ± 0.822414	6.00321 ± 0.760619	**882.821 ± 6.70671**	6.35532 ± 1.26462	879.576±5.6842	2.1412±0.760591
100	2	820.373 ± 5.44482e-05	0.14948 ± 731.704	832.629 ± 0.914018	6.26684 ± 0.961027	**847.165 ± 4.86578**	6.2287 ± 1.26609	840.395±5.46241	1.85868±1.20126
100	3	851.884 ± 5.44482e-05	0.177348 ± 759.789	853.283 ± 0.53162	6.5411 ± 0.455194	**875.194 ± 4.1059**	6.25702 ± 0.265473	871.198±8.04553	1.90601±0.89115
100	4	847.097 ± 5.44482e-05	0.155768 ± 755.538	860.587 ± 1.06527	7.15321 ± 1.82723	**900.594 ± 8.58765**	6.6244 ± 0.998141	887.868±7.41264	2.18509±0.944363
100	5	807.677 ± 5.44482e-05	0.135363 ± 720.391	840.268 ± 1.04452	5.11081 ± 0.339512	**843.654 ± 1.38978**	4.71399 ± 0.547792	829.056±5.0761	1.04715±0.229106
125	1	1030.5 ± 0	0.262764 ± 919.052	1087.26 ± 2.28691	9.07897 ± 1.92937	**1100.67 ± 4.12968**	9.91476 ± 1.56393	1083.17±5.98381	2.51914±0.997763
125	2	984.847 ± 0	0.269806 ± 878.32	1064.96 ± 4.46207	7.75729 ± 1.35058	**1114.02 ± 10.3464**	6.84991 ± 1.04219	1077.42±13.5059	2.29812±0.94873
125	3	1084.62 ± 0	0.297535 ± 967.301	1088.28 ± 0.520548	9.3719 ± 1.07585	**1111.28 ± 8.71383**	8.64589 ± 1.20381	1108.45±8.23354	4.22317±1.71387
125	4	1034.82 ± 0.000108896	0.262211 ± 922.907	1089.77 ± 1.82365	7.31534 ± 0.335313	**1150.75 ± 11.7145**	9.07068 ± 2.10913	1121.5±14.4338	3.96647±1.52282
125	5	1066.71 ± 0	0.265686 ± 951.352	1089.6 ± 0.694753	7.79967 ± 0.678908	**1122.52 ± 14.7049**	9.09817 ± 2.72062	1112.97±12.1717	3.81299±2.06176
150	1	1108.83 ± 0	0.422007 ± 988.787	1209.44 ± 5.0605	8.3708 ± 1.61013	**1221.83 ± 7.61812**	10.579 ± 3.34487	1183.75±13.9869	4.81175±2.17625
150	2	1171.82 ± 0.000108896	0.423958 ± 1044.98	1277.56 ± 5.93505	8.29128 ± 0.929234	**1326.05 ± 11.4681**	8.66003 ± 1.13793	1295.32±11.5507	5.8286±2.30625
150	3	1224.99 ± 0.000108896	0.386116 ± 1092.44	1299.01 ± 6.04625	10.241 ± 2.20703	**1312.21 ± 5.28776**	9.1896 ± 1.3946	1288.98±14.939	4.49413±1.56754
150	4	1091.72 ± 0.000108896	0.387542 ± 973.554	1251.87 ± 5.99892	8.18786 ± 0.553653	**1293.35 ± 11.9353**	9.3645 ± 1.44115	1246.59±21.873	3.78981±1.49482
150	5	1073.19 ± 0.000108896	0.360451 ± 957.048	1198.86 ± 4.54635	9.51866 ± 1.65667	**1238.16 ± 11.4649**	10.2603 ± 1.87435	1201.03±17.9695	4.2936±1.48666
175	1	1309.42 ± 0	0.476003 ± 1167.68	1357.75 ± 5.16512	9.08791 ± 1.48121	**1401.97 ± 13.0728**	8.29186 ± 0.637816	1368.56±13.5855	6.26755±3.41115
175	2	1049.97 ± 0.000108896	0.461774 ± 936.244	1224.7 ± 6.94041	12.0292 ± 3.0892	**1262.87 ± 10.357**	6.93819 ± 0.591516	1219.8±24.1274	9.5612±5.44895
175	3	1242.47 ± 0.000108896	0.491748 ± 1107.94	1292.11 ± 4.14768	8.59071 ± 1.66695	**1345.67 ± 14.8864**	8.36894 ± 1.09874	1311.4±18.0686	5.96779±3.69173
175	4	1195 ± 0	0.598438 ± 1065.5	1308.38 ± 4.68405	9.4354 ± 1.97427	**1316.19 ± 6.40806**	9.56043 ± 1.62796	1274.54±20.8106	6.42413±3.45615
175	5	1154.12 ± 0	0.586398 ± 1029.04	1307.89 ± 8.10066	10.3618 ± 1.82912	**1313.54 ± 9.17159**	12.075 ± 3.14627	1263.64±17.0554	6.47483±3.03541
200	1	1266.61 ± 0.000108896	0.692553 ± 1129.3	1355.08 ± 7.56896	12.1702 ± 4.51101	**1401.55 ± 9.23016**	9.24855 ± 1.62366	1349.36±15.0116	9.80064±3.4197
200	2	1301.33 ± 0	0.704265 ± 1160.26	1402.74 ± 3.57724	9.24602 ± 1.11547	**1439.35 ± 9.97346**	10.4189 ± 1.95214	1409.44±18.8862	9.06333±4.0259
200	3	1246.74 ± 0.000108896	0.558406 ± 1111.69	1371.23 ± 5.47225	9.19963 ± 2.77265	**1419.15 ± 9.37376**	8.7181 ± 1.93453	1392.56±6.74157	9.60859±9.78831
200	4	1192.73 ± 0.000108896	0.656479 ± 1063.42	1298.44 ± 5.05201	9.74109 ± 4.17224	**1314.76 ± 8.01832**	8.57767 ± 1.14974	1265.6±18.5309	12.1087±6.86797
200	5	1345.57 ± 0.000108896	0.667656 ± 1199.76	1409.07 ± 9.16793	10.2477 ± 2.3934	**1432.56 ± 7.33116**	8.83995 ± 0.991145	1419.26±18.9105	11.208±5.38874

Tabela A.4: Resultados para instancias com 50 carros

nPas	Instância	Guloso		Genetico		Genetico+BL		Grasp	
		Z	Tempo	Z	Tempo	Z	Tempo	Z	Tempo
25	1	99.9077 ± 6.80602e-06	0.0097824 ± 89.1168	99.9077 ± 6.80602e-06	1.09144 ± 0.151754	100.112 ± 0.125005	1.05506 ± 0.11467	**102.524 ± 2.53392**	0.0530656 ± 0.0172283
25	2	117.591 ± 1.3612e-05	0.0082054 ± 104.893	**117.593 ± 0**	1.0158 ± 0.0599794	**117.593 ± 0**	0.971461 ± 0.102538	**117.593 ± 0**	0.0446893 ± 0.00698659
25	3	78.8454 ± 0	0.0086125 ± 70.3286	78.8455 ± 6.80602e-06	1.0255 ± 0.0273395	81.3682 ± 3.43766	0.985994 ± 0.144544	**83.0634 ± 3.76289**	0.061124 ± 0.0196656
25	4	91.8701 ± 6.80602e-06	0.0112334 ± 81.9453	91.8701 ± 6.80602e-06	1.3095 ± 0.282637	92.619 ± 2.00418	1.18389 ± 0.100252	**94.0014 ± 2.32857**	0.0594711 ± 0.0173624
25	5	99.8689 ± 0	0.0054809 ± 89.086	99.8689 ± 0	0.822643 ± 0.0412659	**103.931 ± 4.355**	0.819366 ± 0.126471	100.329±0.818856	0.0415078±0.0130748
50	1	341.474 ± 2.72241e-05	0.0367835 ± 304.589	341.881 ± 2.72241e-05	3.04898 ± 0.200278	**355.197 ± 5.83216**	2.7827 ± 0.335249	352.103±6.33322	0.262451±0.130075
50	2	327.75 ± 0	0.0329089 ± 292.349	329.191 ± 2.72241e-05	2.64363 ± 0.409289	335.33 ± 3.12395	2.26112 ± 0.0376379	**342.641 ± 4.13096**	0.218136 ± 0.0479912
50	3	328.986 ± 2.72241e-05	0.0385841 ± 293.447	330.696 ± 0.129135	2.40741 ± 0.065937	341.757 ± 5.31978	2.42706 ± 0.411118	**343.204 ± 5.23943**	0.27419 ± 0.108747
50	4	347.084 ± 0	0.0364133 ± 309.594	349.061 ± 0.00610839	3.07063 ± 0.528295	351.944 ± 3.79381	2.89741 ± 0.613335	**356.494 ± 5.58155**	0.274307 ± 0.0912533
50	5	329.159 ± 2.72241e-05	0.042336 ± 293.598	332.789 ± 0.0106689	3.27478 ± 0.368559	**333.119 ± 0.191496**	3.07487 ± 0.716622	332.44±0.824989	0.363228±0.197179
75	1	583.276 ± 5.44482e-05	0.0950731 ± 520.243	583.38 ± 5.44482e-05	5.59202 ± 0.404125	588.807 ± 3.46843	5.76127 ± 0.997275	**593.048 ± 4.89697**	0.71044 ± 0.185574
75	2	583.631 ± 5.44482e-05	0.0914099 ± 520.563	588.755 ± 0.0326526	4.64241 ± 0.655595	**604.113 ± 3.41735**	4.68811 ± 0.453669	601.992±7.67665	0.662375±0.162456
75	3	552.905 ± 5.44482e-05	0.0971988 ± 493.148	554.52 ± 0.236833	4.92505 ± 1.02178	**556.747 ± 1.70862**	4.37841 ± 0.541339	554.571±1.06476	0.804328±0.4871
75	4	568.948 ± 5.44482e-05	0.112632 ± 507.446	573.797 ± 0.405237	4.47874 ± 0.282366	**584.603 ± 2.97729**	4.03787 ± 0.185838	582.226±2.53926	0.889983±0.482258
75	5	588.058 ± 5.44482e-05	0.0863154 ± 524.517	590.323 ± 0.344937	5.76772 ± 1.1164	**593.717 ± 3.82229**	5.96175 ± 1.5396	589.279±1.59935	0.944917±0.477121
100	1	823.638 ± 5.44482e-05	0.174096 ± 734.594	826.808 ± 0.822939	6.24984 ± 0.767449	848.944 ± 7.99436	7.31442 ± 1.73883	**850.677 ± 6.40845**	1.63565 ± 0.394797
100	2	828.744 ± 0	0.164032 ± 739.158	835.38 ± 1.41204	6.35383 ± 0.351301	**872.951 ± 5.42867**	6.58508 ± 0.987293	864.537±15.2113	1.30372±0.396474
100	3	801.281 ± 5.44482e-05	0.183242 ± 714.642	830.433 ± 0.645176	6.79318 ± 1.26267	**864.798 ± 8.76866**	6.04495 ± 0.789537	855.294±10.8188	1.92132±0.668103
100	4	807.568 ± 5.44482e-05	0.172246 ± 720.26	837.532 ± 0.82508	6.8739 ± 0.443672	**866.424 ± 4.92789**	6.66519 ± 0.671492	854.795±7.61266	1.72854±0.774013
100	5	833.901 ± 0	0.234458 ± 743.696	838.333 ± 0.46701	8.0194 ± 0.749745	867.829 ± 4.85652	9.70635 ± 1.57968	**872.352 ± 10.8756**	2.1397 ± 0.966825
125	1	1072.99 ± 0.000108896	0.334831 ± 956.893	1075.32 ± 0.329447	10.1437 ± 0.688154	**1102.18 ± 6.10841**	10.7582 ± 2.20108	1091.46±8.60714	4.60932±1.32385
125	2	1036.83 ± 0	0.302214 ± 924.664	1072.84 ± 0.705025	10.6395 ± 1.73334	**1114.92 ± 6.99946**	11.4151 ± 2.99307	1106.68±11.7748	3.62789±0.905818
125	3	1047.76 ± 0.000108896	0.371883 ± 934.352	1055.98 ± 0.276577	10.7383 ± 0.748188	**1079.59 ± 3.78122**	10.8767 ± 1.83802	1078.01±5.88471	4.88913±1.86815
125	4	1026.91 ± 0	0.346441 ± 915.775	1077.85 ± 1.16052	11.3361 ± 0.652572	**1112.32 ± 8.04069**	11.0501 ± 2.66356	1100.12±10.6643	2.58964±0.515144
125	5	1073.67 ± 0	0.368946 ± 957.469	1076.79 ± 0.624439	10.3786 ± 0.543683	**1102.94 ± 4.59571**	9.89099 ± 0.621282	1100.15±5.05698	4.49351±2.5181
150	1	1139.59 ± 0	0.474228 ± 1016.18	1238.39 ± 4.34677	10.8628 ± 1.96582	**1250.53 ± 5.58435**	9.68994 ± 1.26838	1226.43±19.8837	5.55151±1.88445
150	2	1273.97 ± 0.000108896	0.506088 ± 1136.03	1292.21 ± 1.11235	14.0669 ± 2.22669	**1348.74 ± 10.9317**	13.9102 ± 3.64175	1336.7±13.591	11.9948±7.08486
150	3	1210.2 ± 0.000108896	0.522475 ± 1079.13	1288.75 ± 6.10462	12.6292 ± 0.812395	**1305.13 ± 5.39713**	12.1617 ± 1.16382	1268.24±12.7338	6.69143±2.72976
150	4	1282.46 ± 0	0.558423 ± 1143.56	1325.07 ± 2.23296	13.2299 ± 0.367776	**1360.27 ± 11.4915**	16.0802 ± 5.87137	1333.91±6.49395	4.9439±2.05296
150	5	1077.42 ± 0	0.471166 ± 960.723	1206.53 ± 3.45508	11.3395 ± 1.68902	**1240.17 ± 7.13049**	9.8805 ± 0.887116	1200.82±15.5351	7.1903±2.46995
175	1	1355.41 ± 0	0.6728 ± 1208.53	1462 ± 6.79732	12.7837 ± 2.09167	**1517.51 ± 7.65372**	12.994 ± 1.69177	1485.23±11.1516	8.28536±3.09752
175	2	1364.19 ± 0.000108896	0.785172 ± 1216.26	1482.99 ± 3.26797	17.2724 ± 6.51036	**1545.52 ± 16.1971**	14.4462 ± 1.79564	1503.4±25.8048	10.1592±5.29967
175	3	1218.92 ± 0	0.71014 ± 1086.74	1381.25 ± 5.22092	9.98468 ± 1.28239	**1415.71 ± 10.9062**	9.8073 ± 1.29196	1394.17±11.7824	6.62684±3.12951
175	4	1269.14 ± 0.000108896	0.949775 ± 1131.33	1361.12 ± 5.56919	14.9124 ± 2.59195	**1401.15 ± 8.3596**	13.2015 ± 2.68747	1367.24±15.0197	7.50721±3.67554
175	5	1381.93 ± 0.000108896	0.542764 ± 1232.31	1468.19 ± 4.75822	10.3519 ± 0.951716	**1503.39 ± 14.6176**	12.5982 ± 2.50468	1485.53±15.7297	7.38116±4.1161
200	1	1537.82 ± 0.000108896	0.863911 ± 1371.09	1636.19 ± 5.06697	15.1012 ± 2.38103	**1647.77 ± 8.35771**	16.4767 ± 4.31613	1625.93±16.6151	14.8304±5.80886
200	2	1503.55 ± 0.000108896	1.1142 ± 1340.29	1662.55 ± 6.51358	18.4346 ± 2.80761	**1733.14 ± 11.3971**	14.9971 ± 0.963936	1678.55±16.3996	14.2927±8.56156
200	3	1549.3 ± 0.000108896	0.93815 ± 1381.26	1652.18 ± 6.64743	17.5634 ± 5.28889	**1725.3 ± 15.0394**	16.0332 ± 1.57454	1661.72±21.0751	8.25517±1.92261
200	4	1592.89 ± 0.000108896	0.845818 ± 1420.23	1623.52 ± 6.96699	16.3891 ± 4.04288	**1664.58 ± 10.6018**	14.2943 ± 2.19447	1641.46±17.5368	14.2915±6.9021
200	5	1591.54 ± 0	0.983583 ± 1418.9	1664.72 ± 5.3368	15.7899 ± 1.77128	**1723.02 ± 7.76345**	15.3554 ± 2.5708	1689.2±17.4383	9.54995±4.22979

B Melhores Resultados Encontrados

Tabela B.1: Melhor resultado para instancias com 20 carros

Instancia	Algoritmo	FObjetivo	passageiros atendidos	Tempo
j_20_25_1	Grasp	183.325	25	0.034705
j_20_25_2	GA + BL	187.929	25	0.726505
j_20_25_3	GA + BL	198.216	25	0.554772
j_20_25_4	GA + BL	195.459	25	0.481888
j_20_25_5	GA + BL	193.812	25	0.577466
j_20_50_1	Grasp	436.822	50	0.107167
j_20_50_2	GA + BL	430.307	50	1.70423
j_20_50_3	GA + BL	450.259	50	0.948247
j_20_50_4	GA + BL	441.577	50	1.29998
j_20_50_5	GA + BL	438.777	50	0.817439
j_20_75_1	GA + BL	619.066	70	1.55122
j_20_75_2	GA + BL	546.087	59	1.67659
j_20_75_3	GA + BL	683.804	73	1.60264
j_20_75_4	GA + BL	597.239	66	1.51499
j_20_75_5	GA	617.587	72	4.34729
j_20_100_1	GA + BL	589.685	65	2.1357
j_20_100_2	GA + BL	699.754	77	1.89683
j_20_100_3	GA + BL	750.917	80	3.49921
j_20_100_4	GA + BL	736.743	80	2.05022
j_20_100_5	GA + BL	671.709	72	1.91243
j_20_125_1	GA + BL	757.471	80	1.90346
j_20_125_2	GA + BL	705.237	76	2.2148
j_20_125_3	GA + BL	693.497	74	2.17945
j_20_125_4	GA + BL	679.153	73	1.80091
j_20_125_5	GA + BL	727.714	78	2.29149
j_20_150_1	GA + BL	748.204	80	1.91246
j_20_150_2	GA + BL	733.281	80	2.32814
j_20_150_3	GA + BL	729.324	80	2.26581
j_20_150_4	GA + BL	718.767	79	2.38076
j_20_150_5	GA + BL	702.619	78	1.86342
j_20_175_1	Guloso	741.706	80	0.175491
j_20_175_2	GA + BL	742.131	80	2.49371
j_20_175_3	Grasp	734.138	78	0.936084
j_20_175_4	Guloso	726.92	80	0.203097
j_20_175_5	GA + BL	711.158	78	3.74121
j_20_200_1	Grasp	754.693	80	2.59501
j_20_200_2	GA + BL	754.132	80	2.83426
j_20_200_3	GA + BL	746.883	80	2.0038
j_20_200_4	GA + BL	728.429	80	3.00917
j_20_200_5	Grasp	745.391	80	1.88527

Tabela B.2: Melhor resultado para instancias com 30 carros

Instancia	Algoritmo	FObjetivo	passageiros atendidos	Tempo
j_30_25_1	GA + BL	149.866	25	0.695725
j_30_25_2	Grasp	158.652	25	0.061021
j_30_25_3	GA + BL	142.062	25	0.707575
j_30_25_4	GA + BL	166.205	25	0.57787
j_30_25_5	GA + BL	170.323	25	0.647292
j_30_50_1	Grasp	411.851	50	0.293198
j_30_50_2	GA + BL	408.592	50	1.7455
j_30_50_3	Grasp	414.651	49	0.192596
j_30_50_4	GA + BL	382.615	50	2.16053
j_30_50_5	GA + BL	429.648	50	2.66625
j_30_75_1	GA + BL	664.848	72	2.47366
j_30_75_2	Grasp	651.297	74	1.24992
j_30_75_3	GA + BL	679.296	75	2.66558
j_30_75_4	GA + BL	639.711	75	2.34313
j_30_75_5	GA + BL	670.425	75	3.50427
j_30_100_1	GA + BL	929.465	100	4.05671
j_30_100_2	GA + BL	909.425	100	4.30861
j_30_100_3	GA + BL	849.271	96	2.8859
j_30_100_4	GA + BL	881.967	95	3.33186
j_30_100_5	GA + BL	913.752	98	2.94832
j_30_125_1	GA + BL	1026.99	111	3.5594
j_30_125_2	GA + BL	1001.92	109	3.67478
j_30_125_3	GA + BL	982.968	107	2.97994
j_30_125_4	GA + BL	1020.75	112	4.07469
j_30_125_5	Grasp	1096.4	120	2.89963
j_30_150_1	GA + BL	1082.47	120	4.0111
j_30_150_2	GA + BL	1075.06	120	4.19611
j_30_150_3	Grasp	1089.68	119	4.10016
j_30_150_4	GA + BL	1107.54	119	4.21592
j_30_150_5	GA + BL	1019.45	111	3.62919
j_30_175_1	GA + BL	1068.02	119	4.39843
j_30_175_2	GA + BL	1108.46	120	5.30337
j_30_175_3	GA + BL	1086.77	119	4.58523
j_30_175_4	GA + BL	1031.48	115	4.41956
j_30_175_5	GA + BL	1121.7	119	5.60891
j_30_200_1	GA + BL	1123.09	120	4.89654
j_30_200_2	GA + BL	1119.5	120	3.95214
j_30_200_3	GA + BL	1120.61	120	4.81063
j_30_200_4	GA + BL	1125.26	119	5.84416
j_30_200_5	GA + BL	1105.58	119	5.2529

Tabela B.3: Melhor resultado para instancias com 40 carros

Instancia	Algoritmo	FObjetivo	passageiros atendidos	Tempo
j_40_25_1	Grasp	127.839	24	0.081835
j_40_25_2	GA + BL	133.744	25	1.35762
j_40_25_3	GA + BL	132.621	25	0.658597
j_40_25_4	GA + BL	134.682	25	0.803355
j_40_25_5	Grasp	127.766	25	0.097273
j_40_50_1	GA + BL	388.078	50	2.18572
j_40_50_2	Grasp	394.254	50	0.161879
j_40_50_3	Grasp	399.57	50	0.191586
j_40_50_4	Grasp	362.15	48	0.505482
j_40_50_5	GA + BL	388.072	50	1.94966
j_40_75_1	GA + BL	626.682	74	4.08147
j_40_75_2	GA + BL	650.032	75	3.90601
j_40_75_3	GA + BL	656.482	75	3.23868
j_40_75_4	GA + BL	618.232	72	4.34623
j_40_75_5	Grasp	686.726	75	1.19279
j_40_100_1	Grasp	891.214	100	1.0952
j_40_100_2	GA + BL	859.188	99	5.03147
j_40_100_3	Grasp	885.378	100	4.29185
j_40_100_4	GA + BL	918.749	100	7.44394
j_40_100_5	GA + BL	845.672	100	5.34821
j_40_125_1	GA + BL	1108.15	125	8.69596
j_40_125_2	GA + BL	1131.08	121	5.98412
j_40_125_3	GA + BL	1125.1	125	9.52052
j_40_125_4	GA + BL	1172.69	125	7.40039
j_40_125_5	GA + BL	1143.22	125	9.42196
j_40_150_1	GA + BL	1234.72	141	8.02341
j_40_150_2	GA + BL	1342.6	148	10.9929
j_40_150_3	GA + BL	1323.18	149	10.8068
j_40_150_4	GA + BL	1314.81	144	11.8487
j_40_150_5	GA + BL	1259.85	139	12.3229
j_40_175_1	GA + BL	1425.88	154	9.91608
j_40_175_2	GA + BL	1284.77	141	8.67652
j_40_175_3	GA + BL	1372.97	147	11.084
j_40_175_4	GA + BL	1327.71	150	12.8071
j_40_175_5	GA + BL	1331.04	151	14.0931
j_40_200_1	GA + BL	1416.4	154	10.4805
j_40_200_2	GA + BL	1464.4	158	8.13019
j_40_200_3	GA + BL	1441.34	154	13.3879
j_40_200_4	GA + BL	1332.53	148	10.1201
j_40_200_5	GA + BL	1448.83	159	11.0512

Tabela B.4: Melhor resultado para instancias com 50 carros

Instancia	Algoritmo	FObjetivo	passageiros atendidos	Tempo
j_50_25_1	Grasp	106.154	24	0.098957
j_50_25_2	GA	117.593	25	1.00225
j_50_25_3	Grasp	87.3882	25	0.064844
j_50_25_4	GA + BL	99.3589	25	1.11886
j_50_25_5	GA + BL	111.579	25	0.771675
j_50_50_1	GA + BL	370.855	50	2.60621
j_50_50_2	Grasp	351.204	50	0.175181
j_50_50_3	Grasp	358.376	50	0.444911
j_50_50_4	Grasp	365.781	50	0.231488
j_50_50_5	GA + BL	333.461	50	5.42481
j_50_75_1	Grasp	603.878	75	0.574334
j_50_75_2	Grasp	623.616	75	0.501521
j_50_75_3	GA + BL	560.255	75	4.04436
j_50_75_4	GA + BL	589.782	75	3.74315
j_50_75_5	GA + BL	602.816	74	4.71029
j_50_100_1	GA + BL	870.482	100	5.31226
j_50_100_2	Grasp	887.543	100	2.2669
j_50_100_3	GA + BL	879.844	100	5.26124
j_50_100_4	GA + BL	875.408	100	8.6689
j_50_100_5	Grasp	888.807	100	1.11839
j_50_125_1	GA + BL	1112.98	125	8.69755
j_50_125_2	GA + BL	1132.02	125	14.9885
j_50_125_3	Grasp	1090.69	125	2.24706
j_50_125_4	GA + BL	1128.7	125	10.1092
j_50_125_5	GA + BL	1112.67	125	9.53011
j_50_150_1	Grasp	1275.32	145	7.45389
j_50_150_2	GA + BL	1370.53	150	11.4871
j_50_150_3	GA + BL	1315.34	148	14.2169
j_50_150_4	GA + BL	1379.42	150	11.8246
j_50_150_5	GA + BL	1248.74	141	9.83979
j_50_175_1	GA + BL	1534.8	167	16.3311
j_50_175_2	GA + BL	1578.53	170	14.9919
j_50_175_3	GA + BL	1437.82	160	8.79105
j_50_175_4	GA + BL	1422.89	156	11.8404
j_50_175_5	GA + BL	1533.57	169	16.5002
j_50_200_1	GA + BL	1664.7	189	20.5298
j_50_200_2	GA + BL	1759.62	190	17.7745
j_50_200_3	GA + BL	1757.45	187	16.828
j_50_200_4	Grasp	1682.56	187	21.9796
j_50_200_5	GA + BL	1740.97	188	13.7621