

Aplicação de Algoritmos Heurísticos na Otimização de Rotas Comerciais em Tempo Real

Jofre Garcia Barros, Fernanda dos Santos Cunha

Curso de Ciência da Computação – Escola do Mar, Ciência e Tecnologia
Campus Kobrasol – Universidade do Vale do Itajaí (UNIVALI) – São José/SC – Brazil

jofrebarros@gmail.com; fernanda.cunha@univali.br

Abstract. *The optimization of commercial routes has become of paramount importance for the companies offering this service, seeking to reduce costs, optimize resources, improve the process and satisfy its customers, and contribute to the improvement of urban mobility. This work presents a computational solution for route optimization in real time, using heuristic algorithms for the best path, structured in two modules (web and mobile) connected via web service. The Google Maps API provides geographic information. The solution was used in a real company and gains were identified regarding route optimization.*

Resumo. *A otimização de rotas comerciais tem se tornado de suma importância para as empresas que oferecem este serviço, buscando reduzir custos, otimizar recursos, melhorar o processo e satisfazer seus clientes, além de contribuir para a melhoria da mobilidade urbana. O presente trabalho apresenta uma solução computacional para otimização da rota em tempo real, utilizando algoritmos heurísticos para o melhor caminho, que está estruturada em dois módulos (web e móvel) conectados via web service. As informações geográficas são obtidas da API do Google Maps. A solução foi utilizada em uma empresa real e identificou-se ganhos referentes a otimização de rotas.*

1. Introdução

O conceito de *smart cities*, ou cidades inteligentes, se define pelo uso da tecnologia para melhorar a infraestrutura urbana e tornar os centros urbanos mais eficientes e melhores de se viver [Tonon, 2010]. As grandes cidades têm crescido cada vez mais, e com isso, os governos, as empresas e as comunidades dependem cada vez mais da tecnologia para superar os desafios dessa rápida urbanização. O que torna uma *smart city* realmente inteligente é o uso combinado de software, infraestrutura de servidor, infraestrutura de rede e dispositivos clientes [Washburn, 2010].

O trânsito se destaca dentre os inúmeros desafios de uma urbanização rápida nas grandes cidades, e este trabalho trata da otimização de rotas para empresas de forma que seus veículos operem por menos tempo nas ruas da cidade. Como benefícios para as empresas que optarem por esse sistema destacam-se a economia dos custos logísticos e de tempo de trabalho, aumento da satisfação dos clientes, possibilidade de atender mais clientes em menos tempo, melhoria na gestão de veículos, entre outros.

Segundo Monteiro (2009) o crescimento do mercado de dispositivos móveis e a criação de novas ferramentas para gerir informações geográficas estão cada vez mais possibilitando o acesso a informações espaciais a qualquer momento e lugar. Dispositivos móveis coletam uma grande quantidade de dados constantemente, seja através do GPS

(*Global Positioning System*), ou do próprio usuário, informando algo sobre determinado local ou compartilhando suas informações em determinado software.

Este trabalho apresenta um sistema web que indica a melhor rota comercial para entregas de uma determinada empresa. O projeto conta também com um aplicativo móvel que alerta o motorista a respeito de eventuais congestionamentos e, fazendo o uso de informações coletadas, alterando a rota em tempo real buscando o melhor caminho.

2. Visão Geral do Sistema

O sistema apresentado neste artigo, possui três módulos distintos: *web*, móvel e *web service*, que interagem entre si.

Na aplicação web, é possível cadastrar as informações dos clientes e calcular a melhor rota. Para calcular a melhor rota utiliza-se o algoritmo genético e o algoritmo de colônia de formigas. Após realizar o cálculo da rota, as informações serão armazenadas no banco de dados da aplicação e posteriormente serão solicitadas pelos outros módulos.

O motorista pode obter as informações sobre as rotas previamente calculadas na aplicação *web*, para isso ele faz uso do segundo módulo da solução, o aplicativo móvel. Este aplicativo faz uso do módulo *web service* do sistema, que fará a comunicação entre os outros módulos. Essa visão é mostrada na Figura 1.

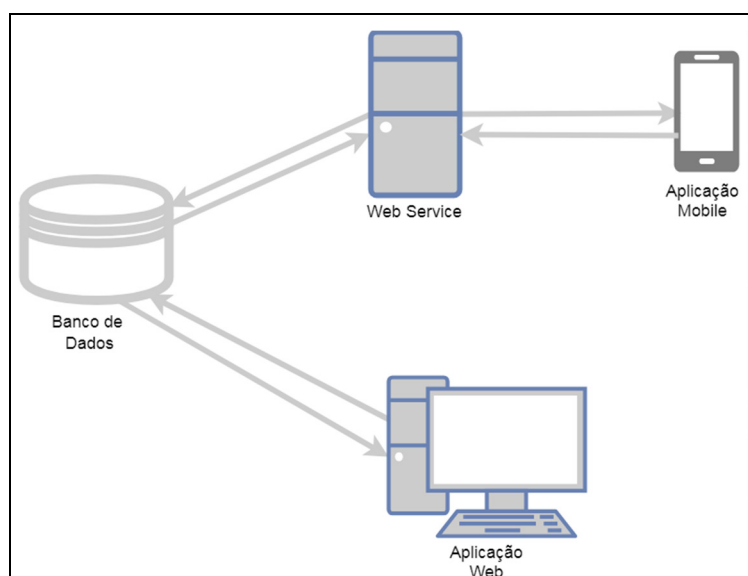


Figura 1. Diagrama da solução proposta.

3. Algoritmos

O problema em questão se assemelha ao problema do Problema do Caixeiro Viajante (PCV). Este problema é um dos clássicos problemas de otimização combinatória, consistido em determinar em um grafo ponderado $G = (N, M)$, denominando por N o conjunto de vértices do grafo e por M o conjunto de arestas, no ciclo hamiltoniano de menor custo [Goldbarg, 2012].

Desta maneira a aplicação dos algoritmos para solução do PCV se adequam para a otimização de rotas. Assim, para a implementação desse projeto, o sistema faz uso de dois algoritmos heurísticos: o algoritmo genético e o algoritmo de colônia de formigas.

Ao gerar uma rota pelo sistema web, ambos algoritmos são executados e aquele que tiver o melhor custo-benefício será armazenado no banco de dados juntamente com a rota. Para verificar o melhor custo-benefício multiplica-se o tempo de execução do algoritmo pela distância total da rota, o algoritmo com o menor resultado terá o melhor custo-benefício. Ao executar a otimização pela aplicação móvel, o sistema deve utilizar somente o algoritmo que contiver o melhor custo-benefício.

Os algoritmos usados foram modelados considerando-se a necessidade execução em tempo real, e buscando atingir a rota ótima o maior número de vezes possível.

3.1. Algoritmo Genético

Algoritmo Genético (AG) é um método de busca estocástica baseado na Teoria da Evolução Natural das Espécies de Darwin (1859), criados por John Holland nos anos 60 [Heinen, 2006]. Usa-se uma população de soluções iniciais, chamadas cromossomos, que através de diversas operações vão sendo evoluídas até que se chegue a uma solução que melhor atenda a algum critério específico de avaliação. Para que isto ocorra, a cada geração os cromossomos são avaliados segundo uma função que mede o seu nível de aptidão, chamada de função de *fitness*. Os cromossomos que tiverem o melhor *fitness* são selecionados para darem origem a próxima geração, através de operações como cruzamentos e mutações. Desta forma, a tendência é que a cada geração o conjunto de soluções vá sendo melhorado, até que se chegue a uma solução que atenda aos objetivos desejados [Goldberg, 1989].

O desempenho do algoritmo genético está diretamente associado à função *fitness*, e neste projeto esta função vai ordenar o caminho de acordo com a distância total. Para atingir a configuração escolhida e fazer sua validação, realizou-se diversos testes, considerando o tempo de execução do algoritmo e o percentual de acerto da rota ótima – dados apresentados na Tabela 1.

Tabela 1. Teste de validação da modelagem do algoritmo genético (valores médios de 25 execuções).

Pontos	Geração	Tempo (milissegundo)	Aproveitamento
10	7	155	98,37%
15	22	832	98,17%
20	38	2710	89,97%
25	64	7538	89,33%
30	65	11786	83,54%

Nesta tabela, com os dados dos testes, pode-se observar o valor médio da geração que o algoritmo encontrou a melhor solução, o tempo percorrido até atingir essa solução e o aproveitamento do algoritmo (o percentual de vezes que o algoritmo chegou próximo da rota considerada ótima). Para a obtenção desses dados foi utilizado a média de 25 execuções em cada ponto e em cada algoritmo. Para a obtenção da rota ótima, foi levado em consideração o melhor resultado obtido em todos os testes.

A população inicial do algoritmo será 100 vezes o número de pontos de entrega/coleta na rota que passará pelo algoritmo, para que possa obter bons resultados com um tempo de resposta aceitável, pois nos testes realizados para a obtenção dos parâmetros do algoritmo foi possível obter um melhor custo benefício com esse número. Da mesma forma que foi possível observar que se obtém um resultado satisfatório

evoluindo a população 3 vezes o número de pontos de entrega/coleta. Durante as gerações o algoritmo deve passar pela seleção, reprodução e mutação.

A seleção será feita por truncamento onde os melhores cromossomos serão selecionados, e o restante será descartada. Esse tipo de seleção pode acabar prejudicando a solução ótima do algoritmo, pois caso exista algum vício nessa população, isso sempre será passado para seus descendentes. Todavia, essa forma de seleção se fez necessária nesse caso, pela velocidade de execução do algoritmo, ao ser comparada com as outras formas de seleção existentes.

O acasalamento que obteve o melhor desempenho, com a seleção escolhida, foi realizar a reprodução do melhor cromossomo com os demais selecionados em 90% dos casos, e no restante simplesmente manter os pais para a próxima geração. Além da reprodução, para tentar diminuir o problema causado pela escolha da seleção por truncamento, é proposto que seja realizada mutação dos genes em 20% dos casos. Em caso de mutação os caminhos existentes no cromossomo devem ser alterados de forma aleatória, gerando caminhos alternativos de forma aleatória e sem possíveis vícios da genética.

3.2. Algoritmo de Colônia de Formigas

Rodrigues (2007) afirma que na metaheurística denominada *Ant Colony Optimization* (ACO), ou Otimização por Colônias de Formigas, existe uma colônia de formigas artificiais que cooperam entre si com o objetivo de encontrar soluções satisfatórias para problemas de otimização discreta. Os algoritmos das formigas foram inspirados pela observação de colônias reais, cujo comportamento é mais dirigido à sobrevivência da colônia do que à sobrevivência de um único componente da colônia.

Para a definição dos parâmetros do algoritmo de colônia de formigas realizou-se o mesmo teste que o algoritmo genético e se obteve os dados contidos na Tabela 2.

Tabela 2. Teste de validação da modelagem do algoritmo de colônia de formigas (valores médios de 25 execuções).

Pontos	Geração	Tempo (milissegundo)	Aproveitamento
10	3	53	100%
15	6	161	100%
20	9	331	100%
25	14	792	100%
30	13	1146	98,55%

Baseado nas informações obtidas no teste desse algoritmo, a quantidade de formigas deve ser 10 vezes superior à quantidade de pontos de entrega/coleta e a quantidade de gerações deve ser 3 vezes a quantidade de pontos, pois com esses valores foi possível obter um melhor custo benefício nos testes realizados. As formigas devem ser populadas inicialmente de forma aleatória, de forma que, cada uma tenha uma rota aleatória. Além delas, é preciso criar um rastro de cada ponto de entrega/coleta a cada outro ponto, que será atualizado levando em consideração a taxa de depreciação ou de evaporação de 50%, pois caso não exista esse fator, todos os caminhos seriam atrativos para as demais formigas.

A cada geração as formigas podem se movimentar nos pontos de entrega/coleta. Para esse movimento é preciso levar em consideração a distância do ponto e a quantidade

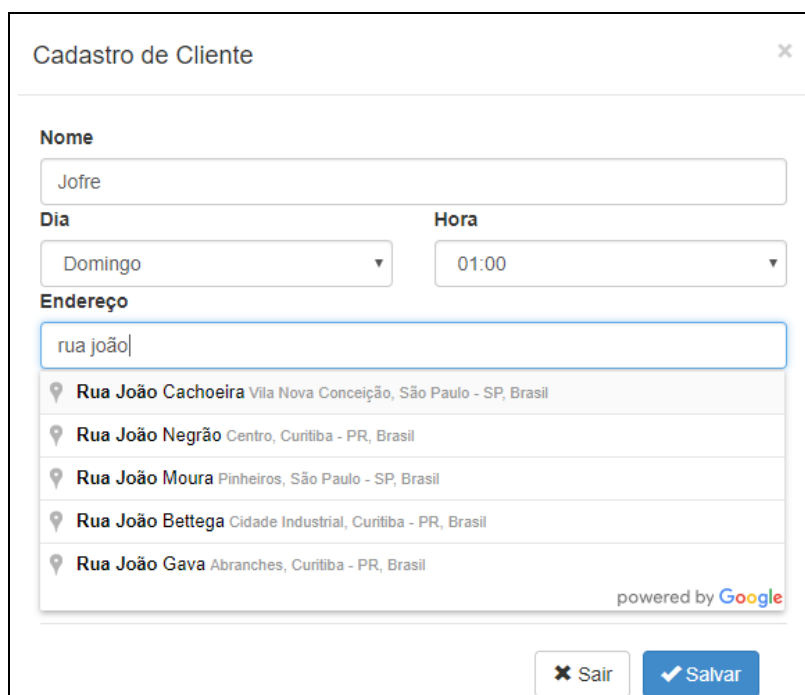
de rastro na ligação desses pontos, ambas as taxas devem utilizar o mesmo peso. Esse processo repete-se em todas as formigas, e até que o número de gerações seja atingido.

4. Portal Web

O *frontend* da aplicação *web* foi desenvolvido via HTML, *javascript* e CSS, com o auxílio da biblioteca *Bootstrap*, por possuir os componentes necessários para o desenvolvimento da aplicação e tornar a aplicação responsiva e com visual mais atrativo. Utilizou-se também *Razor* para a criação das páginas dinâmicas, o que permitiu uma vasta reutilização de código via herança de páginas mestre que continuam conteúdo similar em diversas páginas, necessitando somente da implementação das assinaturas determinadas pela página mestre.

As telas de cadastro foram implementadas para possuir as funções necessárias para o funcionamento do sistema, com pesquisa, inserção, edição e exclusão dos registros. Ao abrir as telas de cadastro como um todo, exibe-se a listagem dos registros inseridos, podendo ser editados ou excluídos. Além disso, há um botão que possibilita a inserção de um novo registro.

Pensando em minimizar a ocorrência de falhas na execução do algoritmo de otimização de rota por não localizar o endereço na API do *Google Maps*, foi adicionado no componente de endereço na tela cliente, o recurso de autocompletar, usando também a API do *Google Maps*, conforme visualizado na Figura 2.



A imagem mostra uma janela de diálogo intitulada "Cadastro de Cliente". O formulário contém os seguintes campos:

- Nome:** Campo de texto com o valor "Jofre".
- Dia:** Menu suspenso com o valor "Domingo".
- Hora:** Menu suspenso com o valor "01:00".
- Endereço:** Campo de texto com o valor "rua joão". Abaixo dele, há uma lista de sugestões de endereços autocompletados:
 - Rua João Cachoeira Vila Nova Conceição, São Paulo - SP, Brasil
 - Rua João Negrão Centro, Curitiba - PR, Brasil
 - Rua João Moura Pinheiros, São Paulo - SP, Brasil
 - Rua João Bettgea Cidade Industrial, Curitiba - PR, Brasil
 - Rua João Gava Abranches, Curitiba - PR, Brasil

Na base da janela, há dois botões: "Sair" (com uma cruz vermelha) e "Salvar" (com um checkmark verde). No canto inferior direito do formulário, há o texto "powered by Google".

Figura 2. Recurso auto completar no endereço.

Ao gerar a rota na tela de Otimização de Rota, o sistema realiza a execução dos dois algoritmos de otimização de rotas (genético e colônia de formigas). Inicialmente a função de obtenção de distâncias fazia o uso da API do *Google Maps*, porém, a grande quantidade de vezes que a distância entre os pontos é solicitada, foi preciso otimizar essa função. Antes de iniciar os algoritmos de otimização de rota, é calculada todas as distâncias possíveis entre os endereços e armazenado as distâncias em memória, com isso

foi possível obter uma grande melhora no tempo de execução dos algoritmos. Ainda foi possível obter uma outra melhora ao armazenar todas as distâncias no banco de dados, e somente consultar a API caso não possua a distância no banco de dados, ou o endereço do cliente seja alterado. Para validar a alteração do endereço do cliente, foi construída uma *trigger* que exclui as distâncias previamente calculadas caso o endereço seja alterado. Após o cálculo da melhor rota os dados são exibidos na tela conforme ilustra a Figura 3.



Figura 3. Tela de Otimização de Rota.

Os dados são exibidos utilizando também a API do *Google Maps* para essa finalidade. Existe também a possibilidade de visualizar os detalhes da rota e imprimir (desenvolvida em *JQuery*).

A implementação do projeto foi feita no padrão de arquitetura de software MVC (*model-view-controller*), dividindo a aplicação em três partes interconectadas. Portanto o *frontend* tem a troca de dados feita com os controladores da aplicação, que por sua vez trocam as informações com o modelo da aplicação, que possui os modelos dos dados, as regras de negócio, a lógica, as funções e a associação com o banco de dados.

5. Aplicativo Móvel

O desenvolvimento da aplicação móvel foi feito utilizando a plataforma de desenvolvimento *Xamarin*. Para essa escolha considerou-se a reutilização do código do projeto *web*, pois o *C#* permite que as classes sejam utilizadas em todos os projetos da solução, por possibilitar a criação do aplicativo multi plataformas, por criar o aplicativo nativo e pela dificuldade na tentativa de implementar a aplicação usando o *Android Studio*, ocasionadas por problemas de ambiente e de emulação do *Android*.

Em contrapartida dos inúmeros benefícios trazidos pela plataforma *Xamarin*, a adaptabilidade com a API do *Google Maps* não é muito eficiente, necessitando de uma implementação manual para realizar a plotagem da rota no mapa conforme a necessidade da aplicação.

O aplicativo é alimentado pelo *web service*., sendo que ele realiza e disponibiliza a rota otimizada e o recálculo da rota. Após retornar a ordem de entrega/coleta dos pontos do *web service*, o aplicativo consulta a API do *Google Maps* para verificar o caminho ponto a ponto para a plotagem no mapa do aplicativo conforme mostra a Figura 4.

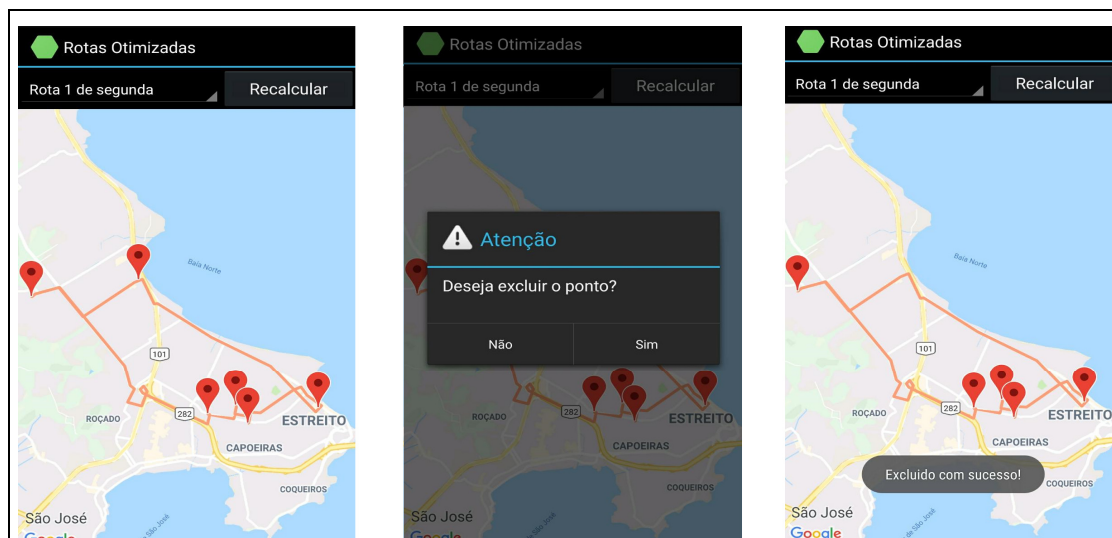


Figura 4. Aplicativo móvel.

6. Web Service

Para realizar a comunicação entre os dois módulos foi necessário o desenvolvimento de um *web service*, desenvolvido em WCF (*Windows Communication Foundation*), o que viabilizou de forma rápida a computação distribuída, por reutilizar as classes dos demais projetos. O *web service* disponibiliza três serviços: a listagem das rotas cadastradas, que popula o *combobox* no aplicativo; a rota otimizada, que retorna a ordem dos pontos otimizados de uma determinada rota; e a rota sem determinados pontos, que recalcula a melhor rota sem os pontos removidos pelo motorista.

7. Resultados

Para avaliação da aplicação, fez-se um estudo de caso de implementação em um comércio de *pet shop* que possui serviço de entrega/coleta de animais, durante 31 dias não consecutivos (de 3^a a sábado, cerca de 7 semanas). Neste estudo de caso a quantidade de pontos de entrega/coleta fica entre um e dez, acrescidos do ponto inicial que sempre será também o ponto final.

Inicialmente os testes objetivaram diminuir a quilometragem percorrida. Esse teste foi realizado na aplicação *web* e resultou nos dados descritos na Tabela 3 (apresentada na página seguinte). A distância da rota manual foi obtida com o responsável do *pet shop* na qual foi gerada com base na experiência do motorista, e a distância da rota otimizada foi gerada pelo sistema.

Ao observar os dados obtidos nota-se que em média se obteve uma melhora de 19,99% na quilometragem das rotas, o que resultaria em uma economia de 3,1 litros de gasolina semanalmente nesse caso, levando em consideração que o veículo utilizado para os testes tem um consumo médio de 10 km/l, o que resultou em uma economia de 161 litros de gasolina no ano.

Tabela 3. Resultado da otimização visando a menor distância.

Nome Rota	Distância Rota Manual (km)	Distância Rota Otimizada (km)	Melhora
terça 1	4,1	2,8	31,71%
terça 2	2,9	2	31,03%
terça 3	2,7	2	25,93%
terça 4	6,9	7,2	-4,35%
terça 6	3,8	2,4	36,84%
terça 7	4,5	1,8	60,00%
terça 8	13,7	9,5	30,66%
quarta 1	2,6	2,1	19,23%
quarta 2	3,2	1,3	59,38%
quarta 3	6,7	6,1	8,96%
quarta 4	4,6	4,5	2,17%
quarta 5	4,2	3,5	16,67%
quarta 6	1,9	1,8	5,26%
quarta 7	3	1,7	43,33%
quinta 1	3,4	3,3	2,94%
quinta 3	8,9	5,3	40,45%
quinta 4	4,2	4,1	2,38%
quinta 6	4	3,2	20,00%
quinta 7	3,2	2,8	12,50%
quinta 8	9	8,5	5,56%
sexta 2	4,3	3,7	13,95%
sexta 3	5,2	3,9	25,00%
sexta 4	8,4	8,5	-1,19%
sexta 5	7,1	4,4	38,03%
sexta 6	2,8	2,4	14,29%
sexta 7	6,8	5,3	22,06%
sexta 8	2	1,1	45,00%
sábado 1	3,9	1,9	51,28%
sábado 2	6	5,6	6,67%
sábado 3	6,7	4,1	38,81%
sábado 4	7,4	5,1	31,08%

Das rotas analisadas, duas obtiverem uma diferença negativa, pois o algoritmo realiza o cálculo da ordem dos pontos, porém o trajeto ponto a ponto é calculado pela própria API do *Google Maps*, e nesses casos os pontos ficaram de forma semelhante à rota manual, porém o percurso acabou sendo um percurso maior.

O segundo teste teve como objetivo diminuir o tempo para realizar as rotas. Esse teste foi realizado na aplicação móvel e resultou nos dados descritos na Tabela 4 (apresentada na página seguinte).

Ao observar os dados obtidos notou-se que em 100% dos casos o tempo de execução da rota melhorou e em média se obteve uma melhora de 25,37% no tempo para a execução das rotas. Esse tempo leva em consideração somente o deslocamento entre os pontos, ignorando o tempo ocioso nos pontos de coleta e entrega.

Tabela 4. Resultado da otimização visando a menor tempo.

Nome Rota	Tempo Rota Manual (min)	Tempo Rota Otimizada (min)	Melhora
terça 1	12	7	41,67%
terça 2	12	8	33,33%
terça 3	9	7	22,22%
terça 4	19	18	5,26%
terça 6	12	8	33,33%
terça 7	15	7	53,33%
terça 8	26	24	7,69%
quarta 1	9	6	33,33%
quarta 2	10	3	70,00%
quarta 3	20	15	25,00%
quarta 4	14	13	7,14%
quarta 5	12	11	8,33%
quarta 6	6	5	16,67%
quarta 7	11	7	36,36%
quinta 1	11	9	18,18%
quinta 3	26	16	38,46%
quinta 4	14	12	14,29%
quinta 6	15	11	26,67%
quinta 7	12	9	25,00%
quinta 8	25	18	28,00%
sexta 2	15	12	20,00%
sexta 3	14	10	28,57%
sexta 4	23	22	4,35%
sexta 5	20	13	35,00%
sexta 6	8	6	25,00%
sexta 7	21	18	14,29%
sexta 8	6	3	50,00%
sábado 1	13	6	53,85%
sábado 2	18	16	11,11%
sábado 3	21	13	38,10%
sábado 4	20	10	50,00%

Com base nos dados coletados, também foi possível identificar que em menos de 10% das rotas o veículo é usado com lotação máxima (o sistema está programado para exibir um aviso quando ainda existe espaço no veículo para inclusão de pacotes na rota). Porém neste estudo de caso em questão nenhuma ação foi tomada pois a otimização do espaço utilizado no veículo causaria ociosidade dos funcionários dependentes da coleta dos pacotes – futuramente pretende-se tratar esta questão no sistema.

8. Conclusão

A partir deste trabalho, identificou-se que os dois algoritmos heurísticos são apropriados ao problema de otimização de rotas por possuir uma alta taxa de conversão para uma solução boa ou ótima mesmo sendo executados em tempo real.

A API de geolocalização do *Google Maps* forneceu todas as informações necessárias para todos os módulos da aplicação, além de ter uma grande usabilidade mesmo de forma gratuita, sendo bem flexível caso exista a possibilidade de ampliar o uso da API.

Quanto a aplicação final, ela atende as necessidades do usuário do tipo administrativo bem como do motorista. Por fim, o *web service* desenvolvido provê uma perfeita ligação entre os módulos web e móvel conforme especificado.

O resultado obtido pela aplicação é considerado significativo ao comparar com os dados anteriores, pois possibilitou uma economia de 19,99% na distância e 25,37% no tempo de execução da rota do estudo de caso. Desta forma, percebe-se que tal aplicação também contribui indiretamente para a melhoria da mobilidade urbana, visto que o motorista consegue elaborar nova rota em frente a congestionamentos, ficando menos tempo no trânsito.

9. References

- Goldbarg, M. and Goldbarg, E. (2012) *Grafos: conceitos, algoritmos e aplicações*. Rio de Janeiro: Elsevier.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.
- Heinen, M. R. and Osório, F. S. (2006) “Algoritmos Genéricos Aplicados ao Problema de Roteamento de Veículos”, In: *Revista Hifen, Uruguiana*, v. 30, n. 58.
- Monteiro, B. R. and Lisboa Filho, J. (2009) “Sistemas de Informação Geográfica Móveis aplicados no Governo Eletrônico Municipal”, In. *I Workshop de Computação Aplicada em Governo Eletrônico WCGE/CSBC. XXIX Congresso da Sociedade Brasileira de Computação. Anais... Porto Alegre*. p.1465-1472.
- Rodrigues, S. B. (2007) “A Metaheurística Colônia de formigas aplicada a um problema de roteamento de veículos: Caso da Itaipu Binacional”, https://acervodigital.ufpr.br/bitstream/handle/1884/12044/disserta%e7%e3o_samuel_bellido_rodrigues.pdf?sequence=1, Junho 2018.
- Tonon, R. (2010) “Cidades Inteligentes”, <http://revistagalileu.globo.com/Revista/Common/0,,ERT338454-17773,00.html>, Janeiro 2018.
- Washburn, D. et al. (2010) “Helping CIOs Understand "Smart City" Initiatives”, <http://c3328005.r5.cf0.rackcdn.com/73efa931-0fac-4e28-ae77-8e58ebf74aa6.pdf>, Março 2018.