FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LEONARDO YOSHIHARU MIYAZAWA

DESENVOLVIMENTO DE UMA SOLUÇÃO PARA ALOCAÇÃO DE TAREFAS COM ROTEAMENTO

MARÍLIA
2016

LEONARDO YOSHIHARU MIYAZAWA

DESENVOLVIMENTO DE UMA SOLUÇÃO PARA ALOCAÇÃO DE TAREFAS COM ROTEAMENTO

Trabalho de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Ricardo José Sabatine

MARÍLIA
2016

MIYAZAWA, Leonardo Yoshiharu

Desenvolvimento de uma solução para alocação de tarefas com roteamento/ Leonardo Yoshiharu Miyazawa; orientador: Prof. Me. Ricardo José Sabatine. Marília, SP: [s.n.], 2016.

59 Páginas.

Trabalho de Curso (Graduação em Ciência da Computação) - Curso de Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília - UNIVEM, Marília, 2016.

1. Roteamento. 2. Otimização. 3. Alocação de Tarefas.

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Leonardo Yoshiharu Miyazawa

Desenvolvimento de uma solução para Alocação de Tarefas com Roteamento.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 9.0 (NOVE)

Orientador:Ricardo José Sabatine _____

1°.Examinador:Renata Aparecida de Carvalho Paschoal _____

2°.Examinador:Allan Cesar Moreira de Oliveira _____

Marília, 06 de dezembro de 2016.

Dedico este trabalho a minha família, por sempre acreditarem em mim, me apoiarem em todas as minhas escolhas e por tudo que fizeram por mim.
À minha namorada, por toda compreensão, companheirismo, amor e carinho.

AGRADECIMENTOS

Aos meus pais, por todo apoio recebido nesses quatro anos de curso e não somente neste tempo, por sempre acreditarem em mim, me confortarem e não me fazerem desistir facilmente das coisas.

À minha namorada, Marina Mazer Gonçalves, por ser essa mulher incrível que me faz bem e me completa, com amor, carinho, conselhos e principalmente paciência.

Ao meu professor e grande mestre Ricardo José Sabatine, que me ajudou muito enquanto fui orientado, que abriu as portas da PERSYS pra aprimorar meu conhecimento, por permitir ter contato com profissionais extremamente dedicados e qualificados e mostrar um mundo completamente diferente do que aprendemos na faculdade.

À minha professora, mestre e conselheira, Giulianna Marega Marques, por todo conhecimento recebido, por ter me recebido muito bem como estagiário da PERSYS e por todas as ideias, conselhos e conhecimento que aprendi.

Aos meus colegas e "professores" de trabalho, Leonardo Lima (mestre de Android e do Back-End), Thiago Corredo Soares (mestre da zoeira e do Front-End), João Paulo Morijo (mestre da zoeira e da Infra) e Éttore Leandro Tognoli (conhecido como Johnny Depp ou Wesley S. ou mestre de Algoritmos Genéticos e Python), agradeço pela paciência em me ensinar um pouco do conhecimento de vocês, pelas zoeiras que não foram poucas e pelo tempo que estive trilhando ao lado de vocês.

Aos meus colegas de sala que sempre me ajudaram quando precisei e me divertiram muito nesses quatro anos de faculdade, espero levar nossa galera pra vida toda se puder.

Aos meus amigos de longa data, desde o pré III/1ª série até hoje, mesmo nos distanciando, alguns que ainda tenho contato, vocês merecem ser lembrados aqui.

A todos os professores do Curso de Ciência da Computação, por contribuírem para minha formação, pela paciência e por ensinarem um pouco do que vocês dominam.

E a todos que puderam contribuir de alguma maneira para que este trabalho saísse, muito obrigado.

“A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro”.

Albert Einstein

MIYAZAWA, Leonardo Yoshiharu. Desenvolvimento de uma solução para alocação de tarefas com roteamento. 2016. Trabalho de Curso. (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino “Eurípides Soares da Rocha”, Marília, 2016.

RESUMO

O princípio básico da otimização combinatória consiste na resolução de problemas de otimização que envolvem função objetiva, conjunto de restrições e relação com variáveis de decisão. A finalidade deste projeto é focar em otimização em relação aos problemas de alocações de recursos e tarefas. O objetivo é apresentar um serviço como solução para o problema de alocação de tarefas, que está diretamente relacionado com otimização de recursos como gasto relacionado ao combustível e tempo, e tarefas como alocar um determinado agente para uma tarefa específica com prioridade, visando encontrar uma solução que resolva o problema com sucesso, cuja finalidade é maximização de desempenho e minimização de custos. Tem-se como contribuição deste projeto o serviço para roteirização que compõe um sistema corporativo cujo objetivo é facilitar e otimizar os gastos como combustível num trajeto de um agente para uma ordem de serviço.

Palavras chave: Alocação de tarefas, roteamento, minimização de custo, otimização.

MIYAZAWA, Leonardo Yoshiharu. Desenvolvimento de uma solução para alocação de tarefas com roteamento. 2016. Trabalho de Curso. (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino “Eurípides Soares da Rocha”, Marília, 2016.

ABSTRACT

The beginning of combination optimization consists on solving optimization problems with objective function, constraints and relation with decision. The area on optimization is quite extensive, then the purpose of this project is focus on optimization of task allocations. The objective is introduce a service as a solution to the problem with task allocation and routing, which is directly related to optimization of resources such as fuel and saving time, and tasks such as allocating a specific agent to a specific task with priority, in order to find a solution that solves the problem with success, which purpose is maximize performance from resources. The contribution of this project is a service for routing which makes up a corporate system whose objective is to facilitate and optimize the expenses as fuel in a route of an agent for an order of service.

Keywords: task allocation, routing, minimizing cost, optimization.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PL	Programação Linear
PI	Programação Inteira
PD	Programação Dinâmica
AG / GA	Algoritmos Genéticos
PM / KP	Problema da Mochila (*Knapsack Problem*)
PCV / TSP	Problema do Caixeiro Viajante (*The Traveling Salesman Problem*)
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol
JSON	Javascript Object Notation
REST	Representational State Transfer

SUMÁRIO

1. INTRODUÇÃO

O problema de alocação de tarefas é um problema comum que as pessoas e principalmente empresas enfrentam diariamente. Por se tratar de um problema que exige a melhor opção nas escolhas de recursos para alocação de determinadas tarefas, nem sempre as pessoas podem resolver sem auxílio de tecnologia. É notável como a tecnologia avança com rapidez e isso reflete na vida das pessoas, trazendo facilidade e conforto que há alguns anos atrás não existia.

O termo “Problema de alocação de tarefas” é genérico e existem muitas variações desse problema. Portanto, este trabalho tem como foco problemas que envolvem roteamento de veículos, chamado de roteirização.

O problema de alocação de tarefas é conhecido por ser um problema relativamente fácil de encontrar, mas não possuir uma resolução simples. Por exemplo, na vida real é possível encontrar problemas de roteamento (ou roteirização) semelhantes a problemas clássicos como do Problema do Caixeiro Viajante (uma variação do antigo Problema do Carteiro Chinês) cujo problema consiste em: dado alguns pontos de entregas, o caixeiro precisa fazer a entrega de maneira eficiente, sem passar pelo mesmo caminho duas vezes e pelo melhor caminho, otimizando tempo necessário para o percurso total e caso esteja num veículo, economizar gasto com combustível passando várias vezes pelo mesmo caminho.

Esse problema conhecido como roteirização pode ser encontrado no dia a dia principalmente devido ao aparecimento cada vez maior de veículos nos centros urbanos, que acaba gerando trânsito em horários de pico, trazendo lentidão e gasto com combustível.

Outros exemplos de problemas de alocação que são comuns partem desde a administração de uma grade escolar, envolvendo professores de determinadas matérias a serem alocados em determinados horários e salas, levando em conta a localização e outras atividades até alocação de agentes com determinada capacidade para tarefas, alocação de funcionários para projetos em empresas.

Neste projeto, a ideia central é criar uma solução que além de otimizar problemas de roteirização e planejamento, solucione problemas com multicritérios. Esses multicritérios podem ser dados como limite de capacidade, prioridade de uma dada tarefa, tempo e algum critério específico num determinado problema, atribuídos com peso indicando a importância da tarefa.

Mesmo estando tão presente e facilmente encontrado no mundo real, o problema de

alocação não é tão simples de resolver como parece. De acordo com Soares (2011), o problema se encaixa na categoria de problemas NP-Difíceis, ou seja, é difícil encontrar uma solução ótima num determinado tempo hábil que pode variar, dependendo do tamanho do problema.

1.1 Motivação e Objetivos

Como motivação, o trabalho aborda os problemas de alocação de tarefas e roteamento, citados anteriormente na introdução.

Os principais exemplos de problemas que serviram como motivação é o problema de roteirização de agentes, onde um agente deve cumprir uma determinada tarefa. Como são várias tarefas distribuídas por várias localidades, o objetivo deste problema é fornecer uma opção de rota a ser percorrida, levando em conta critérios como prioridade de tarefa e distâncias. Outros problemas que foram vistos, mas que não foram estudados a fundo são: alocação de professores em grade escolar e roteamento de veículos.

Dentro desse contexto, o objetivo geral do trabalho é desenvolver um serviço para alocação de tarefas que possa resolver problemas que visam maximização de desempenho e minimização de custos.

O projeto é uma solução para problemas de alocação de tarefas que envolvem especificamente assuntos como roteamento de veículos e alocação de agentes para tarefas, envolvendo recursos como tempo, prioridade de ordem, capacidade do agente para realizar dadas tarefas. Com isso, estaremos tratando também, aspecto de teoria das filas.

Para isso, foi criada uma solução de otimização de tarefas e rotas por meio de um serviço Web. Esse serviço realiza os cálculos relacionados à distância entre pontos utilizando a fórmula de Haversine, através de latitude e longitude dos pontos.

Os objetivos específicos deste trabalho têm foco principal na resolução de problemas de alocação de tarefas com recursos e restrições. Em relação as tarefas relacionadas a recursos, diante de um problema de rotas (roteirização), é ideal buscar a melhor opção de economia de recursos como gasto com combustível e até otimização do tempo. Num problema de roteirização, os critérios que podem alterar a trajetória de uma rota são critérios de tempo (em relação a distância), consequentemente ao utilizar uma rota otimizada, gera menos gasto com combustível. Além disso, existe também a possibilidade de haver problemas que podem surgir diante da rota (problemas na estrada, trânsito, etc.) e a ideia é que o usuário queira fugir desses problemas.

1.2 Organização do trabalho

Estruturalmente, este trabalho é composto por cinco capítulos.

No primeiro capítulo foi feito uma introdução sobre o problema de alocação de tarefas, onde é possível encontrar e vivenciar no mundo real.

No segundo capítulo, foi explicado sobre o problema de alocação de tarefas, onde é encontrado, servindo de motivação para encontrar uma solução e alguns problemas estudados como base.

No terceiro capítulo é descrito sobre a metodologia de solução e sobre trabalho relacionados, onde além de alguns artigos, foi abordado sobre a ferramenta denominada "OptaPlanner", cuja ferramenta estudada é um tipo de solução de otimização que aborda tipos de problemas de alocação de tarefas visualmente e mostrar a solução de como é resolvido.

No quarto capítulo é apresentado o desenvolvimento da proposta de solução do problema de alocação de tarefas para o problema de como foram aplicados alguns métodos para solução do problema.

No quinto capítulo são apresentados os resultados obtidos através de testes no cenário proposto e a conclusão, juntamente dos trabalhos futuros.

2. PROBLEMA DE ALOCAÇÃO DE TAREFA

O problema de alocação de tarefas é uma problemática que representa no mundo real a distribuição de tarefas de maneira ineficiente, consumindo gastos e gerando custos desnecessários. Para resolver este problema complexo é necessário buscar otimização de recursos, melhoria do uso, que pode gerar economia e equilíbrio de recursos utilizados.

Santana (2006) diz que o problema de alocação de tarefas (escalonamento), é extremamente complexo por ser um problema pertencente à classe dos NP-Completos, portanto, qualquer otimização possível de ser realizada é importante.

Em relação aos problemas de alocação, existem problemas que devem respeitar mais de dois critérios, como prioridade, capacidade, além de tempo e localização. O critério de prioridade está ligado à ordem, ou a uma determinada tarefa que tenha prioridade maior. Alguns exemplos que envolvem esse contexto são: distribuição de funcionários numa empresa, dadas características de cada funcionário, horários, tarefas, capacitação de determinado empregado, etc.; Alocação de agentes para tarefas em campo, dada localização do agente e localização da tarefa. Transporte de bens ou pessoas com uso de recursos como veículo, variáveis como tempo, combustível e restrições como limite de tempo, limite da capacidade do tanque; Caixas de diversos tamanhos a serem colocadas em containers de diversas capacidades, etc.

Algumas tarefas de otimização que existem no mundo real podem envolver desde pequenas e médias empresas até grandes projetos de grandes centros industriais e multinacionais. Casos como uma distribuição de funcionários numa empresa; roteirização (rotas que exigem um caminho mais rápido, atualizado e objetivo); determinadas cargas a serem alocadas em diversos containers ou veículos com limites; ou ainda, um sistema de corte de materiais que necessita de uma boa distribuição para minimizar o desperdício e o custo.

Nas diversas empresas, tanto grandes como pequenas (principalmente grandes), um planejamento mal feito pode acarretar em sérios problemas econômicos, gerando desperdícios que poderiam ser evitados. Para realização de um planejamento ideal, deve ser feito uma distribuição otimizada dos recursos para uso (materiais, funcionários, dinheiro, tempo). Os funcionários de uma empresa podem realizar o planejamento básico de algumas tarefas, porém, pode ser ineficiente e levar tempo desnecessário, que poderia ser aproveitado. Para isso, a melhor opção para as equipes de planejamento é planejar a distribuição de tarefas com auxílio computacional, através de ferramentas ou aplicações cuja função é escolher a melhor opção de distribuição de recursos ou tarefas para agentes.

É notável que o problema de planejamento parece simples, ao ponto de ser resolvido por uma pessoa, mas ao aumentar o tamanho do conjunto de variáveis, determinar certas restrições, percebe-se o quão complexos podem se tornar. Cunha (2000) diz que isso é causado pela complexidade do problema de alocação, considerado típicos NP-Difíceis (em inglês, *NP-Hard*), significa que possuem ordem de complexidade exponencial, ou seja, o esforço computacional para a resolução aumenta exponencialmente com o tamanho do problema.

Na tabela 1, que foi construída por Alves (2015), tem como objetivo mostrar exemplos do cenário de alocação encontrado no mundo real, com recursos, restrições e objetivo da alocação. É notável que esses problemas de alocação são facilmente encontrados, porém, o grande problema é alocar todos esses recursos de maneira eficiente, que respeite as restrições e que atinja o objetivo.

Tabela 1 - Exemplos de problema de alocação no mundo real

Objetivo	**Recursos à disposição**	**Restrições possíveis**
Escala de empregados, Funcionários.	- Pessoas; - Tempo; - Capacitação	Carga de trabalho limitada; Turnos devem possuir empregados alocados; Atender preferências de empregados quando possível
Roteirização; Transporte de bens materiais.	- Distância; - Tempo; - Capacidade de carga; - Prioridade de entrega; - Combustível; - Veículo;	Carga deve respeitar o limite do veículo; Consumo de combustível deve ser o mínimo possível; Obedecer ao tempo (não atrasar)
Armazenamento de produtos;	- Veículos; - Containers; - Caixas; - Mochilas;	Carga não pode exceder limite do recurso;

Fonte: ALVES, Marcelo. Resolvendo problemas com OptaPlanner, 2015.

As técnicas e algoritmos para solucionar o problema podem ser expressos matematicamente. O campo da programação matemática é enorme e abrange soluções de áreas como Programação Linear, Inteira e Dinâmica.

Este capítulo aborda sobre os problemas de otimização mais conhecidos, como Problema do Caixeiro Viajante e Problema da Mochila, que serviram de base para realização deste projeto.

2.1. Contextualização do problema de alocação de tarefas

Dados os problemas sobre alocação de tarefas, um problema de otimização pertence à classe matemática, chamada de Programação Linear, cujo objetivo é a função objetiva, que é modelada de acordo com variáveis e restrições das equações e inequações lineares. Porém, caso o problema só assuma valores do conjunto matemático dos inteiros, esse problema também pertence a uma subclasse da programação linear, chamada de Programação Inteira, que serão apresentadas a seguir.

2.2. Problema do Caixeiro Viajante

O problema do caixeiro viajante é um dos clássicos problemas de otimização e será bastante abordado neste projeto. O objetivo do problema é determinar a melhor rota nas cidades realizando entregas percorrendo o menor caminho possível, otimizando custos como transporte e combustível e melhorando o tempo de viagem.

O problema do caixeiro viajante (*The Traveling Salesman Problem*/TSP) tem sido muito utilizado no experimento de diversos métodos de otimização por ser, principalmente, um problema de fácil descrição e compreensão, porém, de difícil solução uma vez que é NP-Árduo e de larga aplicabilidade (PRESTES, 2006).

Goldbarg e Luna (2005) dizem que este problema é um dos mais tradicionais e conhecidos problemas de programação matemática que envolve problemas de roteamento. Esses problemas de roteamento lidam com determinados pontos que são representados por cidades, postos de trabalho, depósitos, entre outros. Neste trabalho, utiliza a ideia do caixeiro viajante como um agente a resolver problemas, passando pelos pontos em determinados locais da cidade ou até do estado.

O problema tem servido de plataforma de teste para investigação de diversas ideias

algorítmicas, porque, além de ser um problema de larga escalabilidade no mundo real, é de fácil compreensão e descrição, mas de difícil resolução por pertencer à classe de problemas NP-Difícil. Por esse motivo, vários métodos foram desenvolvidos com objetivo de resolver instâncias cada vez maiores desse problema e em menor tempo (PRESTES, 2006).

Por ser um clássico problema de otimização, existem muitas formas de solução. De acordo com Goldbarg e Luna (2005), que propõe uma solução por Algoritmos Genéticos, os Algoritmos Genéticos constituem métodos de busca baseados em mecanismos de seleção e evolução natural e os primeiros trabalhos nessa linha são originários de John Holland (1962 e 1970), cujos trabalhos objetivavam replicar os processos utilizados pelos sistemas auto adaptativos em um contexto computacional.

Segundo Lacerda e Carvalho (1999), este problema é considerado NP-Difícil, significa que os algoritmos conhecidos para encontrar a solução exata são intratáveis pelo computador (requer uma quantidade de tempo computacional que aumenta exponencialmente com o tamanho do problema). Os problemas NP-Difíceis podem ser resolvidos com Algoritmos Heurísticos, que podem não garantir sempre a solução ótima, mas que reduz o tempo de processamento encontrando uma solução viável. Portanto, neste trabalho, o algoritmo heurístico que será trabalho é o Algoritmo Genético.

Goldbarg e Luna (2005) relata que grande parte dos algoritmos heurísticos construtivos para este problema utiliza a estratégia denominada “gulosa” ou “míope” de solução. Essa estratégia, em última análise, desenvolve, a cada passo das heurísticas construtivas, a ideia de obter o maior proveito ou ganho possível. Evidentemente, a política global de tomada de decisão pode não comportar sempre ações que levem aos maiores ganhos imediatos. Algumas estruturas matemáticas possuem a propriedade de aceitar uma forma de tomada de decisão míope.

2.3. Problema da Mochila

Este problema é considerado outro clássico problema de otimização que é bastante conhecido, devido à facilidade de entender o contexto do problema (alocar itens ou recursos de vários pesos e valores diferentes numa mochila com peso limitado). O objetivo é preencher a mochila com o máximo de itens, respeitando o limite da mochila e que possuem o maior valor possível, otimizando a alocação. Existem vários tipos de variações para o problema da mochila que serão abordados neste capítulo.

O problema da Mochila é um dos mais importantes e mais intensamente estudados

em Programação Discreta e Otimização Combinatória (MARTELLO e TOTH, 1990). Pode ser descrito de maneira bem simples: uma das variações do problema da mochila é o da mochila limitada, que possui um determinado limite de peso e deverá ser preenchida da maneira mais otimizada possível com itens que possuem pesos variados.

Carvalho (2015) define que o problema da mochila é um problema de programação linear inteira e é classificado como NP-hard, caso alguém deseje listar todas as possibilidades e tomar a maior delas, a complexidade do problema cresce na ordem de 2n, onde n é o número de itens. Supondo um computador que analisa um bilhão de vetores por segundo e retorna a melhor combinação, iríamos precisar de aproximadamente 30 anos para analisar um caso com 60 itens, mais de 60 anos para um caso com 61 itens, conforme Martello e Toth (1990) apresentaram.

2.4. Problema de Roteamento de Veículos

O problema de roteamento de veículos ou problema de roteirização é uma variação dos problemas de alocação de tarefas, sendo um problema clássico de otimização. Em relação as empresas, controlar o gasto com transportes é importante, podendo possibilitar uma redução no tempo de atendimento aos clientes. Escolhendo uma rota aprimorada, gera um melhor uso do combustível gastado pelos veículos de transporte e garantindo um bom atendimento ao cliente devido ao tempo otimizado.

De acordo com Cunha (2000), o termo roteirização de veículos, embora não encontrado nos dicionários de língua portuguesa, é a forma que vem sendo utilizada como equivalente ao termo em inglês, "*routing*", para designar o processo para determinação de um ou mais roteiros ou sequências de paradas a serem cumpridos por veículos de uma frota, com objetivo de visitar pontos dispersos geograficamente, que necessitam de atendimento.

Segundo Laporte et al. (2000), o termo roteirização de veículos consiste em definir roteiros de veículos que minimizem o custo total de atendimento, cada um cada um dos quais iniciando e terminando no depósito ou base dos veículos, assegurando que cada ponto seja visitado exatamente uma vez e a demanda em qualquer rota não exceda a capacidade do veículo que a atende.

Na tabela 2 são citados alguns problemas de otimização e suas variações, mostrando as características de classificação de roteirização, abordando o número de roteiros, qual tipo de localização de clientes, se há limite na capacidade dos veículos, tipos de demandas e números de bases.

Tabela 2 - Classificação de Roteirização Pura

Denominação	**Número de Roteiros**	**Localização de Clientes**	**Limite na capacidade dos veículos**	**Número de bases**	**Demandas**
Problema do Caixeiro Viajante	Um roteiro	Nós	Não	Uma Base	Determinística
Problema do Carteiro Chinês	Um roteiro	Arcos	Não	Uma Base	Determinística
Problema de múltiplos Caixeiros Viajantes	Múltiplos roteiros	Nós	Não	Uma Base	Determinística
Problema de roteirização em nós com múltiplas bases	Múltiplos roteiros	Nós	Sim	Múltiplas Bases	Determinística
Problema de Roteirização em nós com demandas incertas	Múltiplos roteiros	Nós	Sim	Uma Base	Estocástica

Fonte: Adaptado de Bodin et al. (1983, *apud* Cunha, 2000).

Sob a ótica de otimização, os problemas de roteirização de veículos, incluindo o caso particular do caixeiro viajante, pertencem à categoria conhecida como *NP*-difícil (do inglês "*NP-hard*"), o que significa que possuem ordem de complexidade exponencial (CUNHA, 2000).

Cunha (2000) diz que a roteirização de veículos envolve um conjunto muito grande de diferentes tipos de problemas, no qual ele cita que BODIN et al. (1983) apresentaram o

primeiro trabalho que retratava a modelagem de problemas de roteirização e programação de veículos. Os problemas de roteirização podem ser do tipo roteirização pura ou combinados de roteirização e programação. Nos problemas de roteirização pura, condicionantes temporais não são importantes para a definição dos roteiros e sequências de atendimento como coletas ou entregas (CUNHA, 2000).

2.5. Considerações finais

Este capítulo abordou os problemas de alocação de tarefas, alguns problemas mais conhecidos de otimização como Problema do Caixeiro Viajante, Problema da Mochila e o Problema de Roteamento de Veículos. Dentre os métodos matemáticos que envolvem otimização de recursos citados estão: Programação Linear, Programação Inteira e Programação Dinâmica. O capítulo a seguir abordará sobre os diversos métodos de soluções.

3. MÉTODOS DE SOLUÇÃO

Diante dos problemas que envolvem otimização como Problema do Caixeiro Viajante e o Problema da Mochila citados anteriormente, são abordadas diferentes formas e métodos de resolução que serão apresentados a seguir.

3.1. Programação Linear

Os modelos de programação linear constituem um tipo especial de modelos de otimização. Este método é um dos métodos para solucionar problemas de otimização. Goldbarg e Luna (2005) afirmaram que para um sistema ser representado por um modelo de programação linear, deve possuir características como proporcionalidade (a quantidade de recurso consumido por uma atividade deve ser proporcional ao nível dessa atividade na solução final, além do custo de cada atividade ser proporcional ao nível de operação de atividade), não negatividade, aditividade, entre outros.

A programação linear (PL) é um problema de otimização no qual a função objetiva é linear nas incógnitas e as restrições consistem em igualdade linear (LUENBERGER e YE, 2008).

A programação linear trabalha com função objetiva, variáveis e restrições, com objetivo de minimizar ou maximizar. Diante do problema de maximização de lucro e minimização de custo, busca-se a função objetiva, com determinadas variáveis e restrições a serem respeitadas, expressas em equações e inequações lineares.

Um problema de programação linear está em sua forma padrão se tivermos uma maximização da função objetivo e se todas as restrições forem do tipo menor ou igual, bem como se os termos constantes e as variáveis de decisão assumirem valores não negativos (LACHTERMACHER, 2009).

3.2. Programação Inteira

A programação inteira pode ser considerada uma subclasse da Programação Linear, no qual todas as variáveis do problema devem pertencer ao conjunto dos números inteiros, ou seja, devem assumir valores inteiros dentro do problema.

Um modelo de otimização consiste num problema de programação inteira se qualquer variável não puder assumir valores contínuos, ficando condicionada a assumir

valores discretos (inteiros). Ou seja, assume valores inteiros se não puder ser dividido em frações ou valores exatos (GOLDBARG e LUNA, 2005).

Um exemplo clássico de problema de otimização resolvido pela programação inteira é o chamado Problema da Mochila. Esse problema caracteriza-se pelo estreito relacionamento com um grande número de outros modelos de programação. Metaforicamente, entende-se esse problema como desafio de preencher uma mochila sem ultrapassar um determinado limite de peso, otimizando o produto carregado. Foi possivelmente reportado pela primeira vez na literatura por Dantzig (1957) e constitui um marco das técnicas de programação inteira, otimização combinatória e programação dinâmica (GOLDBARG e LUNA, 2005).

3.3. Programação Dinâmica

A programação dinâmica é uma técnica utilizada para a otimização de processos de tomada de decisão. É denominado um processo de decisão “multi estágios” aquele que pode ser desdobrado seguindo um determinado número de etapas sequenciais ou estágios. As alternativas incluídas na conclusão de um estágio são denominadas decisões. A condição do processo dentro de cada estágio é nomeado de estado. Cada estágio inclui a tomada de uma decisão que pode ou não alterar o estado do processo, mas que, obrigatoriamente, representa uma transição entre o estado corrente e o estado futuro do processo (GOLDBARG e LUNA, 2005).

Neste trabalho, o uso da programação dinâmica em relação à otimização gera uma solução determinística, ou seja, mostra uma solução otimizada. O problema é a lentidão na hora de gerar solução, pois os cálculos irão buscar todas as soluções possíveis para mostrar a melhor, algo semelhante ao algoritmo de “força bruta”.

A programação dinâmica é um procedimento de otimização aplicável a problemas solucionáveis usando-se esta sequência de decisões, por sua vez, resultando numa sequência de estados ou caminho e, num valor associado a cada caminho que deve ser minimizado ou maximizado (CARVALHO, 2015).

3.4. Força Bruta

Um dos métodos de solução para problemas de otimização é chamado de algoritmo Força Bruta ou Método Exaustivo e sua principal característica é realizar uma busca detalhada e exaustiva por uma boa solução.

Soares (2011) diz que o algoritmo Força Bruta é um tipo de algoritmo de resolução que aborda diferentes possibilidades de combinações e valores possíveis de forma exaustiva até que encontre uma solução considerada ótima, portanto, esse tipo de algoritmo possui um custo computacional alto e pode levar tempo para encontrar a solução do problema.

Este método pode trazer uma boa solução, já que faz a busca por completo com objetivo de encontrar a melhor solução, porém, pode consumir bastante processamento, já que realiza uma busca exaustiva.

É uma técnica de projeto de algoritmos de fácil implementação, mas que na grande maioria dos casos possui complexidade elevada tendo como importante característica a aplicabilidade em uma grande variedade de problemas. Utiliza abordagem simples para resolver um determinado problema baseando-se diretamente na declaração do mesmo e nas definições dos conceitos envolvidos (HRISTAKEVA e SHRESTHA, 2005).

3.5. Heurísticas

Existem muitas variações do método heurístico para resolver problemas de otimização, já que esse tipo de algoritmo é bastante encontrado como solução. O tipo de heurística mais importante neste trabalho é o Algoritmo Genético. A heurística gulosa e o Algoritmo de Dijkstra podem ser boas soluções diante de um grafo de distâncias, mas não será o foco a implementação dessas soluções neste trabalho.

Uma heurística é uma técnica que busca alcançar uma boa solução utilizando um esforço computacional considerado razoável, sendo capaz de garantir a viabilidade ou a otimização da solução encontrada ou, ainda, em muitos casos, ambas, especialmente nas ocasiões em que essa busca partir de uma solução viável próxima ao ótimo (GOLDBARG e LUNA, 2005).

Bueno (2009) diz que algoritmos heurísticos são métodos que buscam a resolução de problemas através de uma solução viável que se baseia em aproximações sucessivas. Logo, este método encontra as melhores soluções possíveis para problemas, mesmo não sendo soluções perfeitas, pois essa falta de precisão do método não é tratada como deficiência do algoritmo, mas uma analogia ao pensamento humano, uma vez que muitos problemas são resolvidos sem o conhecimento preciso.

3.6. Algoritmo Genético

Neste trabalho, o algoritmo genético será de grande ajuda para gerar soluções a partir do grafo. A grande utilidade deste algoritmo é a certeza de gerar uma solução para o problema, através dos métodos disponibilizados. Porém, a desvantagem do uso é que, mesmo fornecendo as soluções, nem sempre pode obter as soluções "ótimas".

Algoritmos Genéticos são métodos de otimização e de busca, inspirados nos mecanismos de evolução dos seres vivos. Estes algoritmos seguem o princípio da seleção natural e sobrevivência do mais apto, declarado em 1859 por Charles Darwin (LACERDA e CARVALHO, 1999).

Constituem métodos de busca baseados em mecanismos de seleção e evolução natural, onde os primeiros trabalhos nessa linha são originários de John Holland (trabalhos realizados em 1962 e 1970), e tinham como objetivo replicar os processos utilizados pelos sistemas auto adaptativos em um contexto computacional (GOLDBARG e LUNA, 2005).

Os algoritmos genéticos têm características bastante peculiares em relação a outros métodos de busca: baseados em um conjunto de soluções possíveis; Não envolvem modelagem do problema (a modelagem é restrita às soluções); O algoritmo apresenta como resultado uma população de soluções (classificadas qualitativamente pela seleção natural) e não apenas uma; Trata-se de um método probabilístico e não determinístico. (BUENO, 2009).

Na figura 1, Bueno (2009) diz que um algoritmo genético pode ser representado da seguinte maneira:

- Criar população inicial, gerando de forma aleatória com x cromossomos;
- Nova geração é uma etapa para criar uma nova geração (de tamanho fixo (x) ou variável), primeiro: selecionar n cromossomos para reprodução; depois realizar cruzamentos aleatórios;
- Posteriormente permitir que mutações aleatórias ocorram em alguns dos cromossomos gerados com uma determinada chance;
- Selecionar os cromossomos mais aptos para a próxima geração e eliminar os menos aptos no ambiente (análogo à seleção natural, sobrevivência dos mais aptos ao ambiente);
- Substituir os indivíduos que foram eliminados por novos indivíduos.
- Após essa etapa, encaminha para a etapa chamada de Critério de Parada: Caso a solução atenda a um critério de parada, ou caso seja detectada convergência da população, parar. Caso contrário, voltar ao passo de gerar

nova geração e refazer.

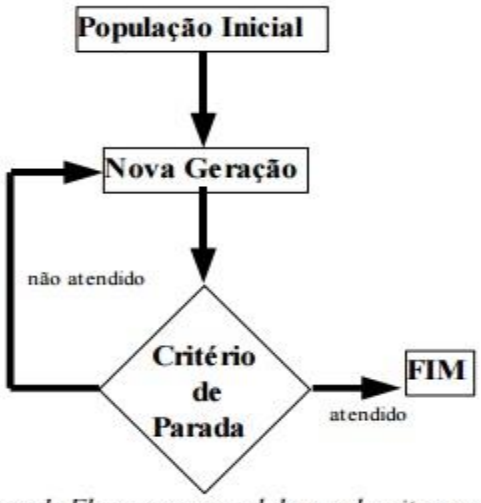

Figura 1: Fluxograma geral de um algoritmo genético

Figura 1 - Fluxograma Básico de um Algoritmo Genético

Fonte: BUENO, Métodos Heurísticos, 2009.

Inspirado no processo de seleção natural dos seres vivos, o Algoritmo Genético seleciona os melhores cromossomos da primeira população (chamada população inicial) para gerar os cromossomos filhos (variantes dos pais) através dos operadores de crossover e mutação. Os operadores de crossover e a mutação são os principais mecanismos de busca para explorar regiões desconhecidas do espaço de busca (LACERDA e CARVALHO, 1999).

O crossover é aplicado a um par de cromossomos retirados da população intermediária, gerando dois cromossomos filhos, onde cada um dos cromossomos pais tem sua cadeia de bits cortada numa posição aleatória, produzindo duas cabeças e duas caudas. Então as caudas são trocadas, gerando dois novos cromossomos (LACERDA e CARVALHO, 1999).

Neste trabalho, o uso do algoritmo foi necessário com objetivo de gerar uma solução, mesmo não gerando sempre a solução considerada ótima. Foi o método escolhido por ser bem referenciado como solução para problemas de otimização, além de possuir uma implementação mais simples e por ser um algoritmo “bioinspirado” na Teoria do Evolucionismo de Charles Darwin.

Baseando-se no método proposto por Lacerda e Carvalho (1999), a figura 2 ilustra a ideia de como o algoritmo funciona neste trabalho. A partir de uma população inicial, seleciona-se os melhores cromossomos ou os mais aptos com o objetivo de gerar

cromossomos filhos destes. Foi aplicado o crossover trocando os genes entre os cromossomos "pais", gerando um novo cromossomo filho com características semelhantes aos pais, mas não idênticas (é possível não realizar o crossover, fazendo com que os filhos sejam idênticos aos pais, preservando alguma solução). Após o processo de crossover, o processo de mutação inicia alterando os genes, caso binários, alterando de 0 para 1 e vice-versa. Neste trabalho não será utilizado binário, mas um outro tipo específico que será melhor explicado no uso dos algoritmos genéticos.

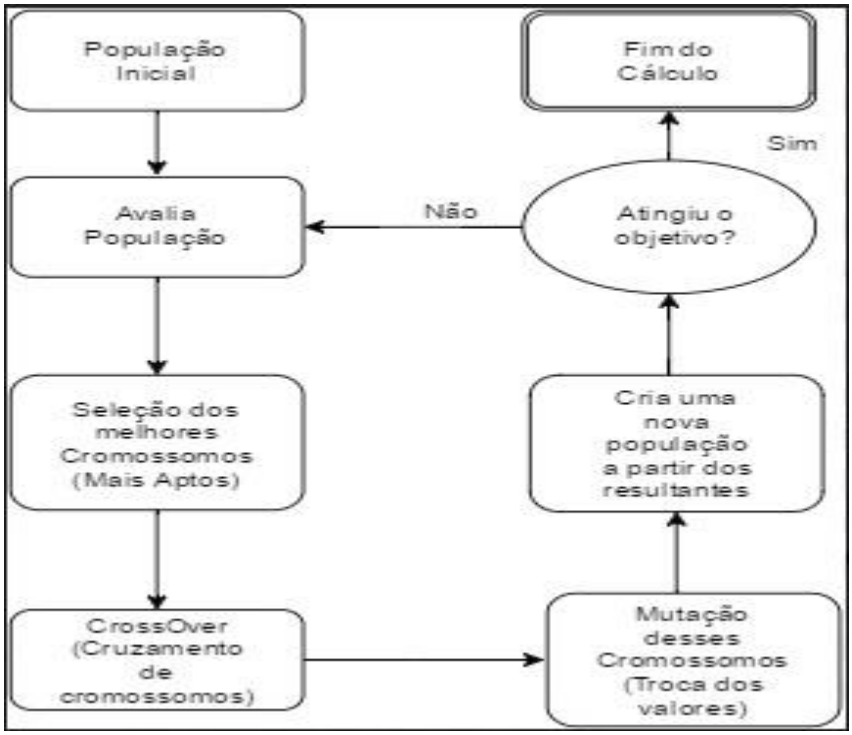

Figura 2 - Funcionamento do Algoritmo Genético

Fonte: Elaborado pelo autor, 2016.

O Algoritmo Genético foi utilizado neste projeto com objetivo de gerar uma solução para o problema de alocação de tarefas. A vantagem do uso deste algoritmo é gerar uma solução para o problema, mesmo que não seja uma solução ótima. Por esta razão, o algoritmo é considerado uma solução determinística.

3.7. Trabalhos relacionados

Neste capítulo, foram analisados alguns artigos e trabalhos sobre os problemas de alocação de tarefas e também será abordado sobre uma ferramenta de resolução de problemas que envolvem otimização de recursos.

Soares (2011) apresentou um estudo como trabalho relacionado ao problema de alocação e os métodos de solução. Utilizou diversos algoritmos para comparar e ver qual se saía melhor em relação ao problema de alocação, comparando os algoritmos "*Branch and Bound"*, Força Bruta, Heurística Gulosa e *"Simmulated Anneling*", traduzido como Arrefecimento Simulado. Nos testes que ele realizou, o *Branch and Bound* forneceu uma solução ótima, porém, que não avaliava todo espaço de soluções devido a uma técnica denominada "*Bounding*", que eliminava as soluções que ao identificar que não era eficiente, já eliminava. Os resultados finais eram que compensava utilizar o algoritmo de Força Bruta, o *Branch and Bound* não otimizado e o Arrefecimento Simulado, que garantiam uma solução ótima. Por fim, ele analisou e verificou que o algoritmo *Branch and Bound* era bom como soluções para instâncias pequenas e médias, enquanto o Arrefecimento Simulado era útil com instâncias maiores.

Videira (2011) realizou uma proposta de solução para o problema de roteirização de veículos utilizando Algoritmos Genéticos, que é uma das várias soluções que podem resolver o problema de roteirização. De acordo com a aplicação e testes realizados, o uso deste algoritmo gerou várias soluções, porém, sem poder determinar quais são ótimas, onde os resultados são satisfatórios dependendo da entrada de dados considerados. O trabalho de Videira (2011) forneceu uma ótima base teórica de algumas coisas, como o uso do Algoritmo Genético para ser utilizado e algumas informações relacionadas a roteirização, além de mostrar uma interface construída bem interessante.

Mauri (2006) mostrou uma nova abordagem para o problema de roteirização de veículos utilizando alguns algoritmos heurísticos e o Arrefecimento Simulado. Ele propôs um modelo matemático e multi objetivo afim de representar e solucionar o problema utilizando Arrefecimento Simulado. Inicialmente, para gerar uma solução inicial, utilizou uma heurística de distribuição responsável pela roteirização dos veículos, formando a agrupação dos locais da rota e também, o sequenciamento de atendimento. Propôs também uso de estruturas de vizinhança, com movimentos de troca que ele diz ser adequada e eficiente.

Uma ferramenta encontrada para resolução de problemas que envolvem otimização de recursos é o OptaPlanner. É uma ferramenta que resolve problemas de alocações

envolvendo otimização. Foi criado como um conjunto de vários algoritmos para resolver determinados problemas, realiza cálculos num determinado tempo hábil procurando a melhor solução possível para o problema, sendo composto por um conjunto de algoritmos de otimização, que propõe resolução dos problemas de planejamento visando o melhor desempenho com menor custo num curto espaço de tempo.

O projeto foi criado inicialmente como "*Drools Planner".* Após a versão "6.0.0", o projeto foi apenas renomeado para OptaPlanner, se tornando independente, mas ainda pode ser opcionalmente combinado com mecanismo de regras do Drools para uma abordagem poderosa para otimização (DE SMET et al., 2013).

Este item foi abordado com objetivo de exemplificar alguns problemas de alocação mais comumente encontrados no dia a dia das pessoas, tais problemas como roteamento de veículos, alocação de professores nas salas de aulas, atribuição de determinadas tarefas para funcionários em empresas, etc.

Toda organização enfrenta problemas de planejamento: fornecer produtos ou serviços com um conjunto limitado de recursos limitados (funcionários, ativos, tempo e dinheiro). O OptaPlanner otimiza esse planejamento para fazer mais negócios com menos recursos (DE SMET et al., 2013).

Na figura 3, retirada da documentação do OptaPlanner, é possível ver a ilustração de vários problemas de otimização citados na ferramenta, desde problemas de roteirização, passando por alocação de equipamentos, de empregados numa grade de horários e empacotamento. A ideia é gerar uma melhor utilização dos recursos e alocar de maneira eficiente em determinado cenário de otimização, como a ferramenta sugere: *"Do more business with less resources"* (Faça mais negócios utilizando menos recursos).

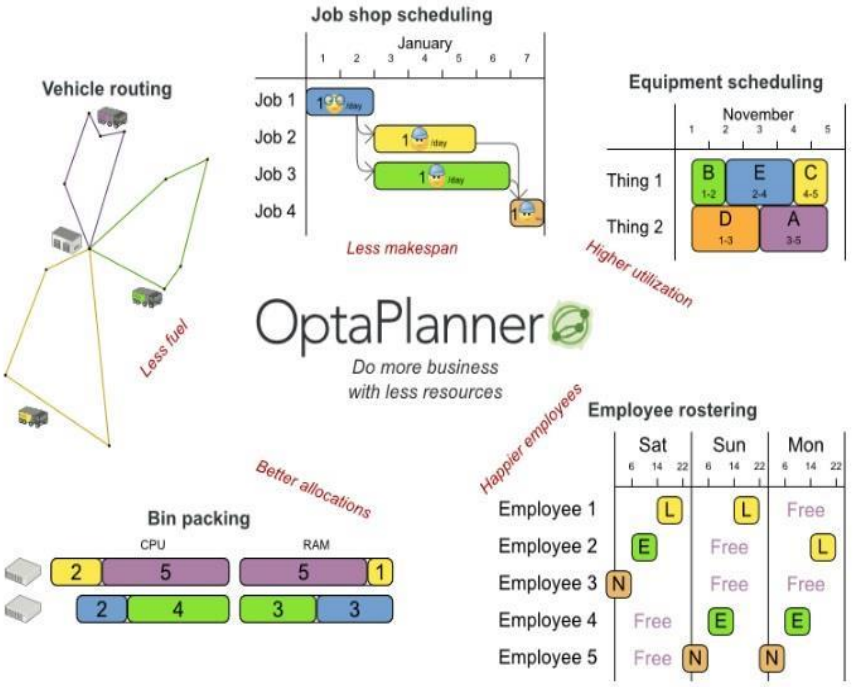

Figura 3 - Ilustração dos problemas abordados pelo OptaPlanner

Fonte: DE SMET et al., OptaPlanner, 2013.

A documentação do OptaPlanner é bem completa e especifica muito bem o que a ferramenta é capaz de fazer. De acordo com De Smet *et. al.* (2013), o problema de planejamento é considerado NP-Completo, que significa que são problemas de difícil resolução. As duas técnicas de resolução comuns não são suficientes: um algoritmo de força bruta é considerado bom, mas que leva muito tempo; um algoritmo rápido irá retornar uma solução que é longe da ideal.

De Smet et. al. (2013) diz que normalmente um problema de planejamento tem, pelo menos, dois níveis de restrições: uma restrição negativa de nível complexo que deve ser respeitada; Exemplo: um professor não pode ensinar duas aulas diferentes ao mesmo tempo. Uma restrição menos complexa e negativa, que deve ou não ser respeitada se isso puder ser possível. Por exemplo: um professor que não gosta de ensinar na sexta-feira à tarde. Alguns problemas têm restrições positivas também: Uma restrição positiva (ou recompensa) deve ser cumprida, se possível. Por exemplo: um professor que gosta de ensinar na segunda-feira de manhã.

Problemas que envolvem planejamento são comuns em empresas tanto de pequeno porte até de grande porte, pois envolvem principalmente os funcionários, que são responsáveis pelo andamento dos projetos e negócios. Cada funcionário tem uma determinada

capacitação de tarefas, onde essa capacidade pode acarretar problemas como sobrecarga ou ociosidade de um funcionário se não houver um planejamento correto.

Mas esses problemas de planejamento tendem a ter um número incrivelmente grande de possíveis soluções, porém, muitas dessas soluções são inúteis. Uma solução viável é uma solução que não quebrar quaisquer restrições negativas. Há casos que não há soluções viáveis. Cada solução viável é uma solução possível. Uma solução ideal é uma solução com a maior pontuação. Os problemas de planejamento tendem a ter uma ou mais soluções ótimas. Há sempre, pelo menos, uma solução ótima, mesmo no caso em que não há soluções viáveis e a solução ótima não é viável. A melhor solução encontrada é a solução com a pontuação mais elevada determinada por uma aplicação em um determinado período de tempo. A melhor solução encontrada é provável que seja viável e, com o tempo, é uma solução ideal (DE SMET et al., 2013).

O OptaPlanner especificamente trabalha com questões caracterizadas por objetivos explícitos, os quais devem ser alcançados utilizando recursos limitados e que respeite as restrições do negócio. No menu principal, é possível identificar quais exemplos o usuário da ferramenta OptaPlanner pode simular.

3.7.1. Exemplo do Problema de Roteamento de Veículos no OptaPlanner

Este item aborda o problema de roteamento de veículos mostrado pelo framework "OptaPlanner". Nele, o problema de roteamento de veículos consiste em: numa frota de veículos, pegar os objetos de cada cliente e trazê-los para o depósito. Cada veículo pode servir vários clientes, mas tem uma capacidade limitada e consequentemente, deve passar por todos eles sem repetição de rota (DE SMET et al., 2013).

Na figura 4 é mostrado um esboço de como acontece o roteamento de veículos, partindo de um ponto inicial (nesse caso, depósito), onde vários veículos devem passar pelos pontos de entrega/atendimento de uma maneira otimizada, gastando o mínimo de combustível possível.

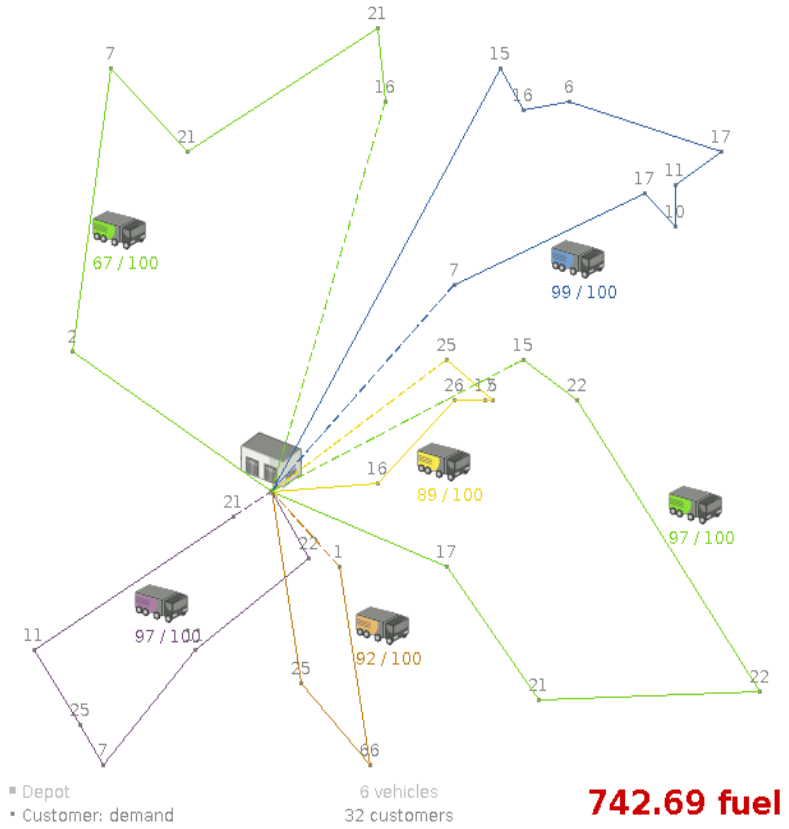

Figura 4 - Exemplo de Roteamento de Veículos no OptaPlanner

Fonte: DE SMET et al., OptaPlanner, 2013.

O problema é típico de roteirização, onde dados alguns pontos de entrega no mapa, é necessário passar por todos eles sem repetição de rota, podendo percorrer o menor caminho possível, otimizando recursos como tempo e combustível ou percorrendo de maneira a gerar ganho de tempo maior, mesmo com risco de gastar mais combustível que pelo menor caminho. Há varias restrições nesse problema: a capacidade do veículo é limitada e não pode carregar muitos itens; tempo de viagem pode ser longo; o principal a se fazer é minimizar a distância, para otimizar combustível de todos os veículos (DE SMET *et al.*, 2013).

Muitas empresas enfrentam um problema de roteamento de veículos diariamente: determinar a melhor ordem para entregar uma série de itens para o número de locais utilizando uma frota de veículos. Eles enfrentam este problema para reabastecer suas lojas, para entregar produtos, oferecer serviços aos clientes, para cumprir a manutenção em seus dispositivos, etc. (DE SMET et al., 2013).

No mundo real, o problema de roteamento de veículos possui muitas suposições falhas. Encontrar uma boa solução é difícil: não há atalhos. É preciso ser capaz de otimizar sem fazer suposições. No entanto, não é possível percorrer todos os estados possíveis de uma

forma de força bruta, mesmo em relativamente pequenos problemas, devido a limitações de hardware consome muitos recursos computacionais. Então é necessário utilizar algoritmos flexíveis, como as heurísticas aplicadas no OptaPlanner para resolver casos de maior dimensão (DE SMET et al., 2013).

3.7.2. Exemplo do Problema de Alocação de Funcionários em Projetos no OptaPlanner

Este problema consiste num cenário real que as empresas enfrentam em relação à planejamento de tarefas em projetos e problemas a serem resolvidos pelos funcionários de uma empresa.

Agendar todas as tarefas em tempo e modo de execução é necessário para minimizar atrasos no projeto. Cada trabalho é parte de um projeto. Um trabalho pode ser executado em diferentes formas: em cada sentido é um modo de execução que implica uma duração diferente, mas também diferentes usos de recursos (DE SMET et al., 2013).

O problema de alocação de funcionários num projeto possui várias restrições: precedência de trabalho (um trabalho só pode começar quando todos os trabalho anteriores terminarem); capacidade de recursos (não utilizar mais recursos que os disponíveis); Recursos são locais (compartilhada entre trabalhos do mesmo projeto) ou global (compartilhado entre todos os trabalhos); Recursos são renováveis (capacidade por dia) e não renováveis (capacidade para todos os dias); minimizar a duração de cada projeto; minimizar a duração de toda a programação multi projetos (DE SMET et al., 2013).

3.8. Considerações Finais

Neste capítulo foram abordados possíveis métodos de solução e uma ferramenta para resolução do problema de alocação de tarefas, denominada como OptaPlanner. Dentro dele, há vários exemplos que envolvem generalizações e problemas que necessitam de uma distribuição otimizada. Essa ferramenta mostra visualmente como funciona a resolução dos problemas de alocação de tarefas e um algoritmo sendo executado. Também foi relatado alguns trabalhos relacionados ao tema deste projeto. No capítulo a seguir, será mostrado como o projeto tratará o problema de alocação e mostrar uma possível solução.

4. ESPECIFICAÇÃO E DESENVOLVIMENTO DA SOLUÇÃO PROPOSTA

O projeto tem como objetivo fornecer uma solução para otimizar alocação de tarefas e roteamento de tarefas. A solução do problema de alocação visa encontrar o menor caminho possível, resolvendo de maneira otimizada e sem repetição de caminho.

Para criar a solução, foi escolhido utilizar o Algoritmo Genético. A seguir, será abordado especificação do problema de alocação de tarefas e como o mesmo será solucionado de maneira otimizada.

4.1. Cenário Proposto

Por ser um tema extenso, existem muitas variações do problema de alocação. Dentro deste problema, o cenário escolhido para estudo é o problema da roteirização utilizando o problema do caixeiro viajante como base. Simplificando o objetivo, é gerar a melhor rota visando otimizar o gasto com combustível e consequentemente otimizar o tempo (em relação a distância). Além desse cenário, o trabalho tem como objetivo escolher a melhor opção de alocar os agentes de acordo com a capacidade deles pra determinadas tarefas, como por exemplo, alocar o agente mais capacitado para uma tarefa considerada difícil ou alocar o agente menos qualificado para uma tarefa mais simples.

Para isso foi desenvolvido um micro serviço para otimização de rotas, implementado num sistema corporativo (será focado no próximo capítulo), que será melhor especificado a seguir.

4.2. Proposta de Solução

A solução tem como principais funcionalidades:

- Georreferenciar os endereços das tarefas;
- Calcular distância entre pontos com baixa precisão (em linha reta) ou com alta precisão (levando em consideração vias de locomoção);
- Listar as distâncias de um endereço (ponto) para um grupo de pontos (matriz de distancia);
- Otimizar a rota de uma agente (técnico em campo) com base em prioridade e sua localização atual;

- Para um grupo de agente atribuir uma tarefa da forma mais otimizada possível;
- Para um grupo de agente atribuir uma tarefa com base em um horário especifico;

Os principais requisitos não funcionais:

- Utilizar mais de uma API para resolver um endereço para geolocalização, com isso, aumentar a tolerância a falha do sistema;
- Ser um serviço agnóstico;
- Ser uma API RESTFul;
- Ser assíncrono.

4.3. Arquitetura da solução

Foi criado um serviço web que vai otimizar os problemas que envolvem roteirização e alocação de tarefas. A arquitetura da solução baseia-se no conceito de micro serviço. Segundo Fowler (2014), o estilo arquitetural de micro serviços pode ser definido como uma abordagem ao desenvolvimento de uma aplicação como um conjunto de pequenos serviços autônomos, cada um rodando em seu processo e comunicando-se através de mecanismos como uma API oferecida via HTTP.

Os micro serviços fazem parte dos sistemas atuais de empresas, oferecendo vantagens e desvantagens em relação à arquitetura anterior, conhecida como Monolítica. O cenário a ser estudado e implementar este trabalho será abordado posteriormente no próximo capítulo. Segundo Fowler (2014), aplicações monolíticas podem ter sucesso, mas um número crescente de equipes de desenvolvimento estão se frustrando com elas, especialmente quando um número crescente de aplicações precisa ser disponibilizado na nuvem. Mudanças são acopladas, fazendo com que alterações em uma pequena parte da aplicação exija um novo build e *deploy* do monolito inteiro. Escalar o monolito significa escalar a aplicação como um todo, ao invés de escalar somente as partes que requerem mais recursos.

Micro serviços são pequenos e focados em realizar uma única tarefa. Eles são limitados de acordo com os limites do negócio. Isto faz com que se torne óbvio onde um trecho de código específico reside. Ao manter o foco nos limites do serviço, o crescimento desnecessário do serviço é evitado junto com todas as implicações que isto pode trazer (NEWMAN, 2015).

Aplicações monolíticas também se tornam difíceis de entender e modificar, especialmente quando elas começam a crescer e novos membros precisam ser adicionados ou substituídos nas equipes de desenvolvimento (NAMIOT, 2014).

Abaixo, uma imagem que representa a diferença entre arquitetura monolítica e arquitetura de micro serviços:

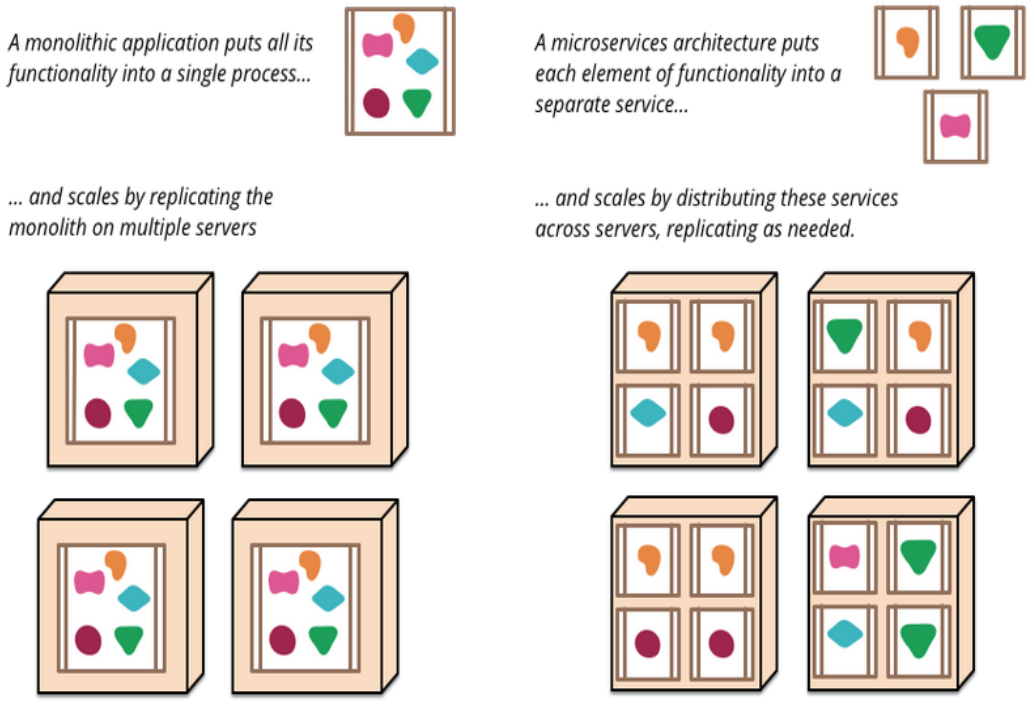

Figura 5 - Diferenças de Arquitetura Monolítica e Micro Serviços

Fonte: Fowler, Microservices, 2014.

Nesta imagem, é possível ver algumas diferenças: a arquitetura monolítica coloca todas as funcionalidades num único processo, que acaba sendo vantagem para realizar *deploy*. Mas a grande desvantagem é concentrar vários módulos dentro da aplicação, correndo o risco de cair todo o sistema se algum desses módulos falhar. A vantagem do uso dos micro serviços é justamente a desvantagem da arquitetura monolítica, caso algum módulo falhe, não há risco de todo sistema parar.

A grande base de código presente nas aplicações monolíticas reduz a produtividade, diminui a qualidade do código, acaba com a modularidade e impede que desenvolvedores trabalhem de forma independente. Times inteiros precisam coordenar os esforços para desenvolvimento e *deploy* (NAMIOT, 2014).

As principais entidades (serviços) da arquitetura do cenário são:

a) API’s para resolver geolocalizaçao: Para esse serviço foi selecionado o serviço do

Google do Google Maps e o serviço do OpenStreetMap. São dois serviços que fazem quase a mesma coisa, a ideia de utilizar esses dois serviços é aumentar a tolerância a falhas do sistema. Por exemplo, se o Google estiver fora do ar, é possível realizar a busca utilizando o serviço do OpenStreetMap. O serviço do Google Maps é bem conhecido, gratuito e de fácil uso por parte do usuário. Desenvolvido pela equipe do Google, é um dos serviços especializados disponibilizados para qualquer pessoa utilizar. Como é bem conhecido, muitas aplicações, sites e dispositivos utilizam o serviço para GPS ou buscar rotas e estradas e é difícil conhecer alguém que nunca utilizou ou até mesmo ouviu falar. O serviço disponibilizado oferece possibilidade do uso de GPS, obter rotas, distâncias e latitude e longitude de determinado ponto.

O serviço do OpenStreetMap é semelhante ao do Google Maps, oferecendo uma interface convidativa ao usuário para percorrer livremente observando fotografias aéreas, possibilidade de utilizar GPS assim como o Maps do Google. Foi desenvolvido por uma comunidade voluntária de pessoas que mapeiam e que contribuem para manter atualizados os dados sobre estradas, trilhas, etc. Através do uso desse serviço, assim como o Google Maps, é possível obter os dados de rotas, distâncias e até latitude e longitude de determinado ponto no mapa.

b) Micro serviço de localização: Esse micro serviço é responsável por monitorar os técnicos em campo para determinar a otimização da rota em caso de uma alteração imprevista na agenda do dia. Armazena todos os dados dos agentes em campo, então através desse serviço implementado, é possível monitorar em tempo real onde cada agente está executando suas tarefas. Além disso, pode ser consultado quais tarefas esse agente executou, em quais datas e horários, foi realizado.

c) Micro serviço para roteirização e alocação de tarefas: esse serviço é o próprio trabalho, possuindo ligação direta com o micro serviço citado no item anterior. Será melhor abordado nos itens a seguir e no próximo capítulo, em relação a implementação no cenário e testes.

4.4. Calculando a diferença entre dois pontos

Para realizar o cálculo da distância entre dois pontos, foi utilizado a “fórmula de Haversine” cujo objetivo é calcular a distância entre dois pontos sabendo-se a latitude e longitude de cada ponto. Essa fórmula é uma equação utilizada para navegação, onde dada latitude e longitude de dois pontos, através dessa equação, encontra-se a distância entre um

ponto ao outro. É possível encontrar duas maneiras de calcular a distância, podendo ser por alta precisão (fazendo uma consulta via API) ou baixa precisão.

A figura 6 ilustra a definição do problema, dados latitude e longitude dos pontos iniciais, finais e até outros pontos que estão entre eles e é necessário saber a distância entre esses dois pontos.

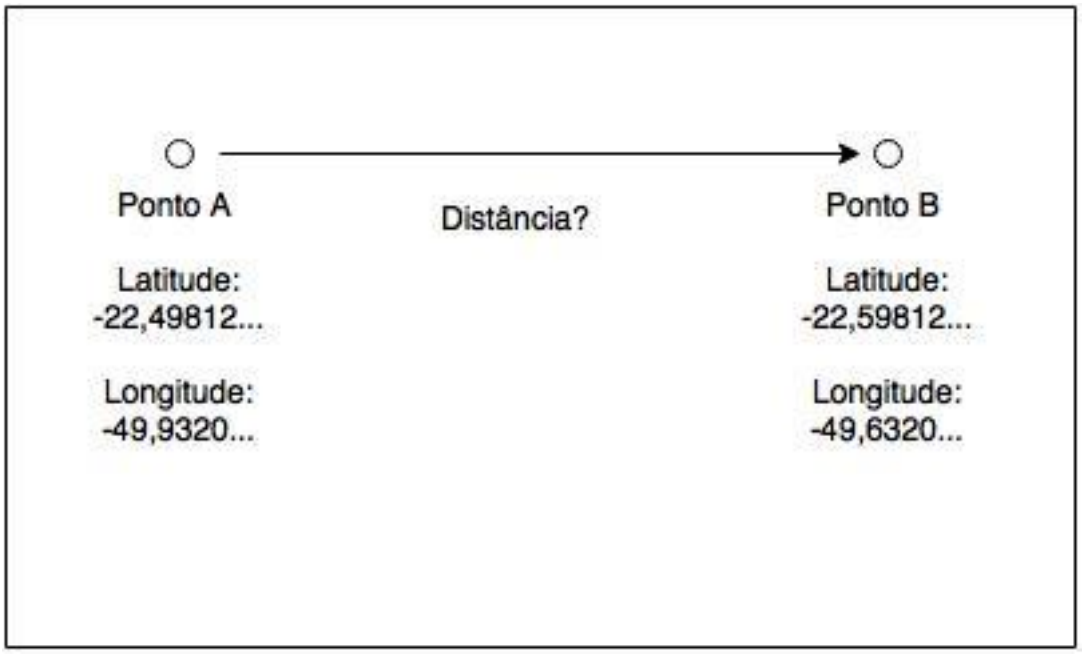

Figura 6 - Representação de distância entre dois pontos

Fonte: Elaborado pelo autor, 2016.

A ideia é, sabendo-se a latitude e longitude do ponto inicial e do ponto final, utilizar Haversine para calcular a distância entre esses dois pontos. Esse cálculo matemático foi considerado de baixa precisão, de acordo com testes realizados neste projeto. Segue abaixo uma imagem da fórmula de Haversine:

$$d = 2r\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right)+\cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\psi_2-\psi_1}{2}\right)}\right)$$

Figura 7 - Equação de Haversine

Fonte: CHOPDE e NICHAT, 2013.

Na figura 7 foi mostrada uma representação da equação de Haversine, onde d é a distância entre dois pontos com longitude e latitude (ψ, φ) e r é o raio da Terra.

É dito de “baixa precisão” devido o cálculo não levar em conta certas rotas como o Google faz. Pode ser dito que o cálculo é realizado em “linha reta”, pois o cálculo pode indicar distâncias que não tem um distanciamento entre dois pontos com bastante precisão. Em relação ao serviço do Google, o cálculo leva em conta rotas altamente precisas (visto que a vista é realizada por satélites, imagens aéreas e outros), trazendo uma certeza bem alta da distância entre certos pontos.

4.5. Gerando a matriz de distancia

Através dos pontos de origem, pontos a serem atendidos (percorridos) e o ponto final (destino), é possível representar as distâncias calculadas pela equação de Haversine anteriormente através de uma matriz de distâncias. Essa matriz de distâncias é montada através da relação entre os pontos a serem percorridos, incluindo o ponto inicial e o destino final.

A figura 8 mostra como foi representada as distâncias entre os pontos:

	Destino	Ponto A	Ponto B	Ponto C
Origem	0	35,67	49,85	96,73
Ponto A	165,89	0	74,83	67,53
Ponto B	83,75	75,92	0	55,22
Ponto C	35,93	67,53	55,22	0

Figura 8 - Exemplo de Matriz de Distâncias

Fonte: Elaborado pelo autor, 2016.

Na figura 8 está representado uma matriz de distâncias, que compara distância entre os pontos iniciais, pontos a serem percorridos (A, B, C) e ponto final (destino). Observa-se que a diagonal principal da matriz é zerada, pois representaria a distancia de um ponto para ele mesmo, além de não poder ligar origem ao destino sem passar obrigatoriamente por todos os outros pontos.

É possível criar um grafo ligando cada um desses pontos de acordo com a matriz,

simulando a conexão entre cidades. Por exemplo, fazendo uma comparação com o TSP (Problema do Caixeiro Viajante), partindo do ponto inicial e passando por cada cidade até o ponto final (destino), pode ser representado segundo a figura 9:

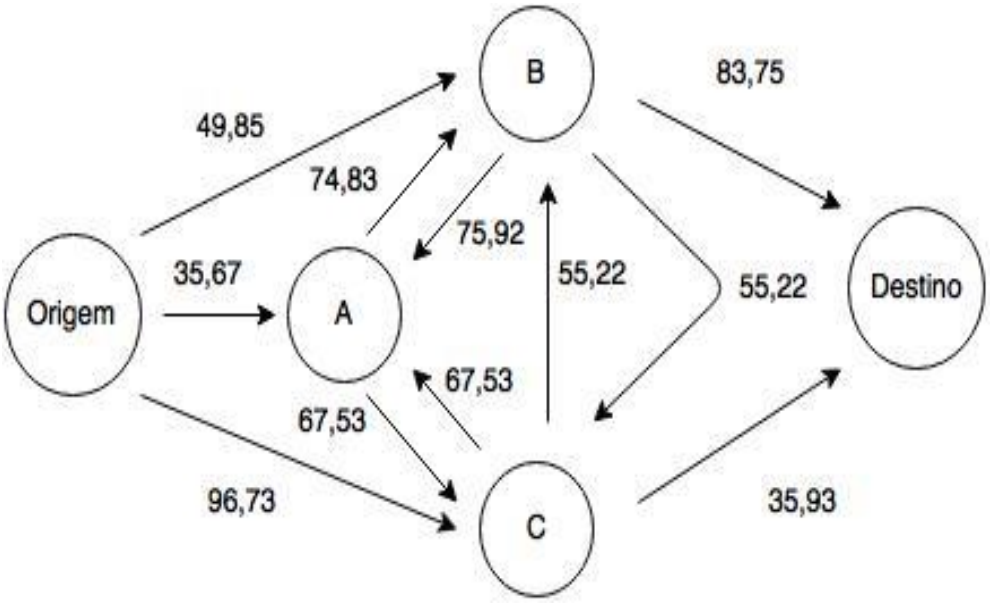

Figura 9 - Exemplo de Grafo de Distâncias

Fonte: Elaborado pelo autor, 2016.

A figura 9 mostra a representação de um exemplo em forma de grafos da matriz de distâncias que foi citada anteriormente. A ideia é fornecer opções para percorrer o grafo de uma cidade a outra, cada um com uma distância entre um caminho ao outro, com objetivo de passar por todos os pontos até o destino final.

4.6. Algoritmo Genético usado na Roteirização

Neste trabalho, o método escolhido foi o Algoritmo Genético. Para isso, foram primeiramente criadas as classes que representarão as principais propriedades do algoritmo genético, como o Crossover, Mutação e Seleção de indivíduos.

Na figura 2 que ilustra um fluxograma de um algoritmo genético, pode ser utilizado como base para explicar como ele funciona neste trabalho.

- Dada uma população inicial, geralmente, é criada de maneira aleatória, cabendo ao usuário definir o tamanho, por exemplo, iniciar com 200 indivíduos, sendo todos gerados randomicamente, sem repetição de genes. Um indivíduo é composto por um conjunto de genes. Um gene possui a

representação cromossômica formado binariamente (por zeros e uns) ou por outro “material”. Nesse caso, o material são as tarefas alocadas com uma determinada sequencia, onde essa sequencia mostra um possível caminho a ser percorrido. Por exemplo, uma sequência de tarefas pode ser representada como “ABCDE”, indicando os pontos passando pelo A, depois B, depois C, e assim por diante. É necessário não repetir tarefas de forma a otimizar o problema de roteamento, passando por todas as tarefas/pontos.

- Após a etapa da população inicial, deve ser avaliada essa população gerada aleatoriamente. Encontra-se uma das etapas mais importantes do processo, no qual o algoritmo gera uma nota de acordo com a função de aptidão de cada gene. Nesse trabalho foi utilizado o critério de atribuir a maior nota para os genes que são formados pelos genes que possuem o valor das tarefas o menor possível, mostrando possuir um bom trajeto (supondo que uma tarefa pode ser representada como: “ABCDE”, isso indica os pontos a serem passados, quanto menor o valor das distâncias entre cada ponto, mais otimizado é a rota), além de atribuir as maiores notas para aqueles genes que tem um atendimento prioritário de ordens (nesse trabalho, para um agente que executa ordens de serviços, atender os pontos mais emergenciais denominados como prioridade emergencial antes de um ponto que tem uma prioridade normal é mais importante, por isso também recebe uma nota mais alta).
- Depois de avaliar a população, o algoritmo seleciona os indivíduos que possuem os genes mais aptos (maior nota em relação à representação do menor trajeto e do atendimento prioritário).
- Após selecionar os indivíduos dentre toda população que foi gerada, é aplicado o método de Crossover, que é o cruzamento dos genes. A ideia de realizar o crossover é selecionar dois indivíduos que serão chamados de “pai” e “mãe”, e gerar um novo indivíduo, com alguns genes de cada um deles.
- Aplicado Crossover, em seguida, é aplicada mutação que é representada por uma taxa relativamente baixa (1%), devido aos riscos de perder algum gene importante e gerar um indivíduo que não seja melhor ou igual ao anterior. Na codificação binária, basta trocar um dígito sendo zero ou um. Nesse trabalho poderia ser representada a troca de letras que representam pontos/tarefas para gerar um indivíduo diferente.

- Após todas essas etapas, chega-se a conclusão de obter um indivíduo, cuja estrutura cromossômica é formada por um gene que fornece uma solução para o problema de roteamento.

A finalidade de utilizar o algoritmo é a certeza de gerar uma solução para o problema de alocação de tarefas com roteamento. Uma das estratégias de solução desenvolvidas, além do algoritmo genético (seleciona os mais aptos), pode calcular distância entre dois pontos utilizando a Fórmula de Haversine e imediatamente criando um grafo; outra estratégia que pode fazer é realizar uma consulta no serviço do Google utilizando uma chave de permissão. Caso não houver uma dessas chaves, ele irá calcular por Haversine.

4.7. Roteirização dos agentes com base em prioridade

Esse recurso da solução tem como objetivo receber uma lista de pontos de sua prioridade (alta, normal e média), como também o ponto de origem e o ponto final.

As restrições são:

- Tarefas com prioridades alta devem ser executadas primeiro, independente do custo e assim por diante;
- O ponto final é opcional, sendo somente o ponto de origem obrigatório.

No capítulo 5, será mostrado alguns testes e a execução da roteirização de agentes de acordo com prioridade de tarefas.

4.8. Considerações Finais

Este capítulo abordou as formas de resolução do problema de alocação de tarefas que tem foco nos problemas de roteirização, utilizando como base o Problema do Caixeiro Viajante. Foi abordado sobre a arquitetura da solução, citando o cenário proposto, além de propor uma solução utilizando os Algoritmo Genéticos.

Nesse capítulo foi falando a fundamentação de como resolver utilizando algoritmos genéticos e seus métodos para escolher os indivíduos mais aptos.

Além do cenário, foi citado como pode ser representado as distâncias entre os pontos a serem percorridos, formando uma matriz de distâncias que estão relacionadas e um grafo de distância, montado a partir dessa matriz.

5. RESULTADOS OBTIDOS

Diante do problema de alocação, existem várias soluções que podem solucionar ou até mesmo resolver e otimizar o problema. Neste trabalho, foi escolhido o uso do algoritmo genético devido a fácil implementação e comparação a Teoria do Evolucionismo de Darwin (seleção dos mais aptos). Teve como resultado o desenvolvimento de um serviço para roteirização e alocação de tarefas para agentes dada uma determinada ordem.

Neste capítulo são abordados os resultados da implementação de uma solução para o problema de alocação de tarefas com foco no roteamento de veículos, utilizando o Problema do Caixeiro Viajante como base.

5.1. Aplicação dos testes de Roteirização ordenados por Prioridade

Foram aplicados alguns testes para verificar o funcionamento do serviço utilizando o POSTMAN, que é uma aplicação gratuita que simula um cliente, com objetivo de testar API's. Posteriormente será abordado sobre a aplicação implementada num sistema corporativo.

O Postman foi utilizado para gerar os testes iniciais da aplicação em relação a roteirização de agentes. Nele, foram testados passar o ponto de origem, destino e distribuídos alguns pontos a serem percorridos.

Para realizar os testes iniciais, foi utilizada a comunicação via HTTP utilizando o verbo "POST" para enviar o ponto inicial (origem), ponto final (destino) e entre esses dois pontos, foram criados vários pontos que serão percorridos cada um com uma determinada prioridade de 1 ao 3 (Normal, Alta e Emergência, representados pelas cores respectivas: Azul, Amarela e Vermelho).

Na figura 10 é ilustrado como foram passados origem e destino e mais alguns pontos a serem percorridos no caminho. Cada ponto foi informado o nome, a localização via latitude e longitude desse ponto e o nível de prioridade.

```
{
    "startingPoint": [-22.2169572, -49.6565663],
    "finalPoint": [-22.22011959, -49.6453875],
    "tasks": {
        "persys": {
            "location": [-22.232840, -49.971797],
            "priority": 3
        },
        "tauste_sul": {
            "location": [-22.2258641, -49.9331484],
            "priority": 1
        },
        "prefeitura": {
            "location": [-22.21992224, -49.94784033],
            "priority": 2
        },
        "rodoviaria": {
            "location": [-22.230566, -49.922686],
            "priority": 1
        },
        "univem": {
            "location": [-22.231948, -49.968098],
            "priority": 3
        },
        "unimar": {
            "location": [-22.235222, -49.971328],
            "priority": 1
        },
        "hering": {
            "location": [-22.21790565, -49.95034591],
            "priority": 1
        }
    }
}
```


Figura 10 - Exemplo de envio no POSTMAN com prioridades

Fonte: Elaborado pelo autor, 2016.

Após o preenchimento destes campos, é enviada a requisição acima e obtida a seguinte resposta, mostrando em ordem decrescente, dos pontos de maior prioridade para os de menor, representada pela figura 11:


```
[
  {
    "task": "univem",
    "sequence": 0
  },
  {
    "task": "persys",
    "sequence": 1
  },
  {
    "task": "prefeitura",
    "sequence": 2
  },
  {
    "task": "hering",
    "sequence": 3
  },
  {
    "task": "unimar",
    "sequence": 4
  },
  {
    "task": "tauste_sul",
    "sequence": 5
  },
  {
```


Figura 11 - Exemplo de Resposta ordenada por Prioridades

Fonte: Elaborado pelo autor, 2016.

5.2. Cenário testado para utilização

O cenário escolhido para teste do serviço desenvolvido foi sistema corporativo denominado eProdutiva, utilizado pela empresa PERSYS. Em contato com a empresa, foi autorizada o uso da marca e de seu software para o desenvolvimento e teste deste trabalho, sua autorização é apresentada em anexo ao final deste trabalho.

O sistema da eProdutiva é composto por recursos como Ordem de Serviço, cuja finalidade é o controle de ações realizadas pelos usuário. Por exemplo, os próprios desenvolvedores e funcionários administrativos utilizam para relatar com o que estão trabalhando, o agendamento dessa ordem (tarefa) levará quanto tempo para ser finalizado, entre outros. Além disso, o uso da Ordem de Serviço foi essencial para testar o serviço desenvolvido neste trabalho, juntamente do cadastro de agentes para realização de determinada tarefa.

O serviço foi testado no sistema móvel do eProdutiva. Para testar o serviço em funcionamento, é necessário o usuário do eProdutiva móvel obter acesso ao sistema web. Por exemplo, um chefe controla as ações de seu funcionários através da ordem de serviço, que mostra as tarefas que o funcionário deve executar. Basta o chefe agendar o endereço da tarefa a ser realizada, agendar data e hora a ser realizada pelo agente “João” (agente criado como exemplo) e salvar. O sistema web imediatamente sincroniza com o serviço móvel, chegando uma notificação no celular do “João”. Obs.: É necessário o agente ter conexão com internet, para imediatamente alertar o agente da tarefa a ser realizada.

5.3. Funcionamento e aplicação da ordenação de tarefas

Dado algumas ordens de exemplo criados no sistema web do eProdutiva e atribuídas para o agente que está conectado no sistema móvel, as ordens chegarão inicialmente por data e hora de agendamento, como demonstrado no item anterior.

Num determinado dia pode haver várias ordens para o agente realizar, cada uma com uma prioridade específica. As prioridades podem ser: Normal, Alta e Emergência.

É possível ordenar essas ordens de acordo com a prioridade de cada uma. Na figura 12, ao lado do ícone de agenda no canto superior direito, existe um ícone que executa a tarefa de ordenar e alocar de maneira eficiente as tarefas de acordo com algum critério. Foi definido que as ordens vão respeitar primeiramente o critério de prioridade de ordem e logo após esse

critério, vem o de distância (mesmo que as ordens estejam todas com diferentes prioridades, ao ordenar por prioridade, também leva-se em conta a distância, afim de otimizar). Depois, caso todas as ordens tenham a mesma prioridade, o segundo critério é distância a partir da localização do agente até cada tarefa. A figura 12 mostra as ordens com diferentes prioridades e diferentes distâncias:

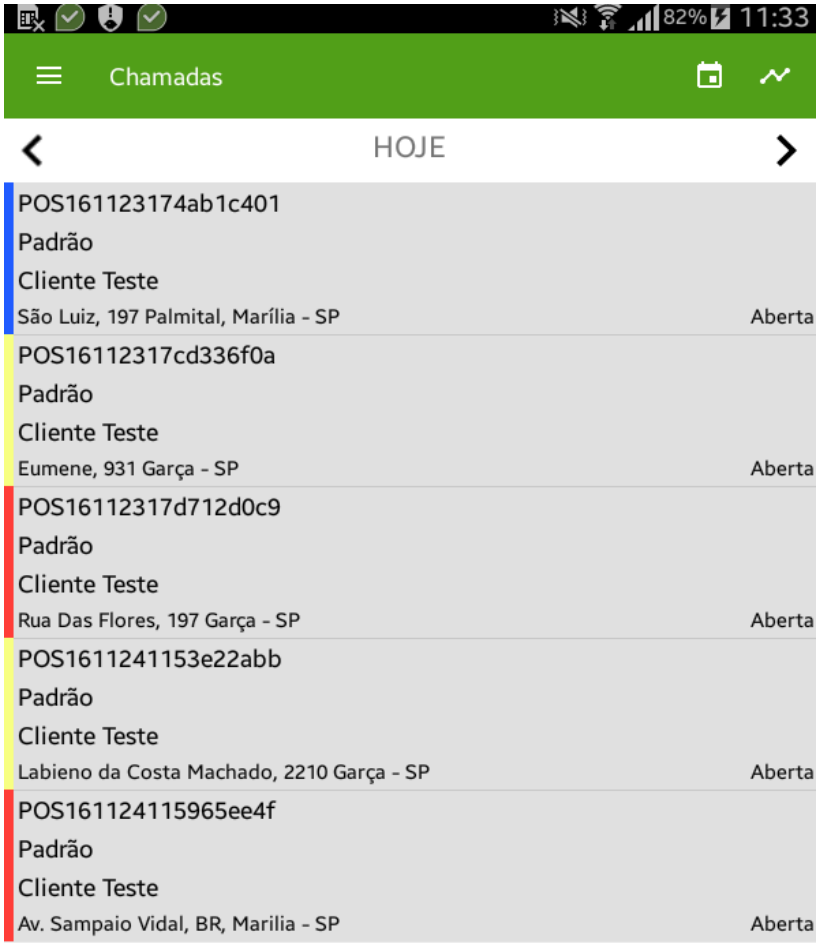

Figura 12 - Imagens das ordens designadas ao agente

Fonte: Elaborado pelo autor, 2016.

Na figura 12, é possível ver os detalhes de cada ordem e que são separadas por cores que indicam prioridade de cada ordem. Em relação às cores, o vermelho significa a prioridade emergencial daquela ordem; em relação ao amarelo, significa que a prioridade da ordem é Alta; por fim, as ordens de cor azul mostram prioridade normal. Para ordenar as ordens por prioridade, basta o agente/usuário clicar no ícone ao lado do ícone da Agenda e aguardar.

Na figura 13, é possível ver as ordens de maneira ordenada de acordo com as prioridades de cada ordem. Nesse caso, o critério de distância não foi ignorado, mas colocado

como segundo critério, gerando assim a lista ordenada por prioridade de ordem e distância. O teste foi realizado a partir da localização atual do agente, neste caso, o teste foi realizado dentro da UNIVEM (Avenida Higino Muzi Filho).

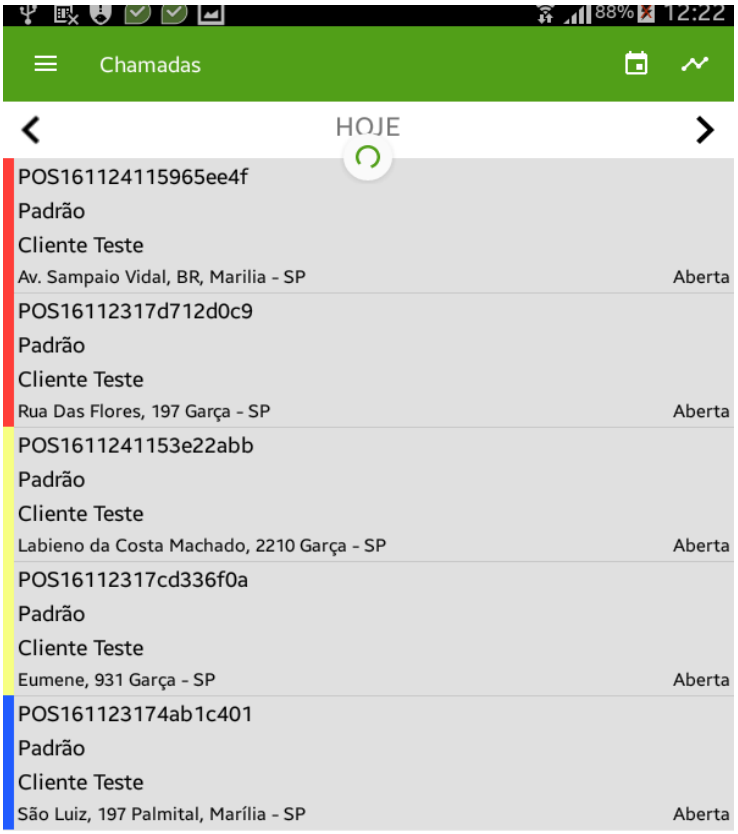

Figura 13 - Ordens após organização por prioridade

Fonte: Elaborado pelo autor, 2016.

É possível ver como as ordens trocaram de posição de execução, ordenadas pelo nível de prioridade de cada uma.

Caso as ordens sejam todas com a mesma prioridade, a ordenação acontece pelo critério de distância. Na figura 14, é mostrada como as ordens serão atendidas mesmo todas possuindo a mesma prioridade de atendimento.

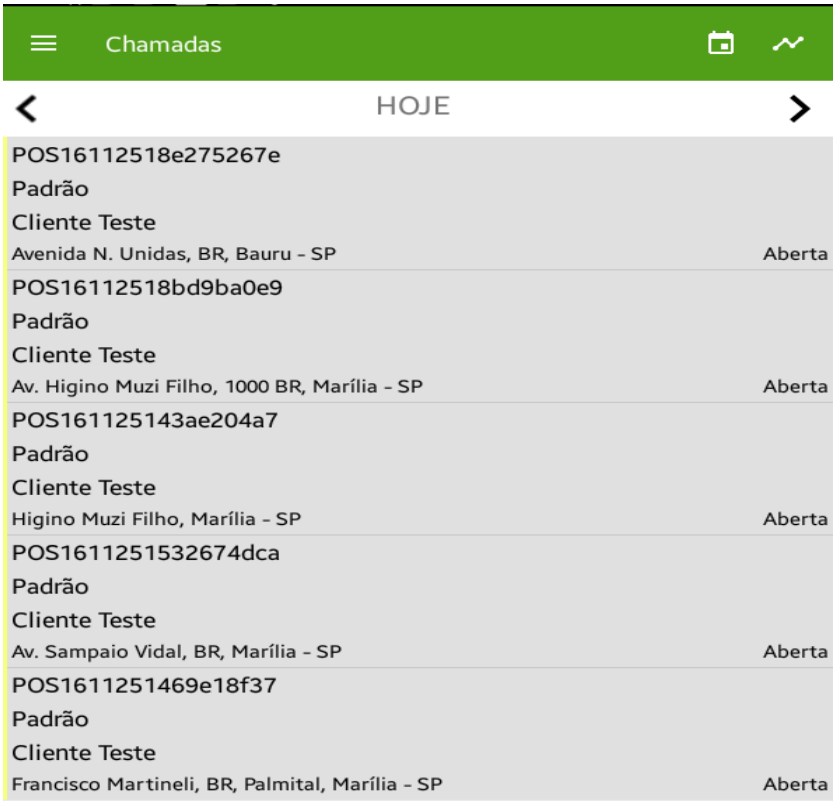

Figura 14 - Exemplo de Ordens de mesma Prioridade

Fonte: Elaborado pelo autor, 2016.

Na figura 14, é notável que as ordens estão fora da ordenação por distância (a primeira delas está localizada em Bauru). O teste foi realizado a partir do endereço da UNIVEM (Av. Higino Muzi Filho). Após clicar no ícone ao lado da agenda no canto superior direito, o sistema carrega e reorganiza por distância, visto que todas as ordens possuem a mesma prioridade (alta).

Na figura 15, é visto que as ordens que foram organizadas por distância, partindo do endereço da UNIVEM, já que todas tem a marcação das ordens de cor amarela, neste caso, significando prioridade Alta. Por ser todas de mesma prioridade, o critério que ordenou foi a distância.

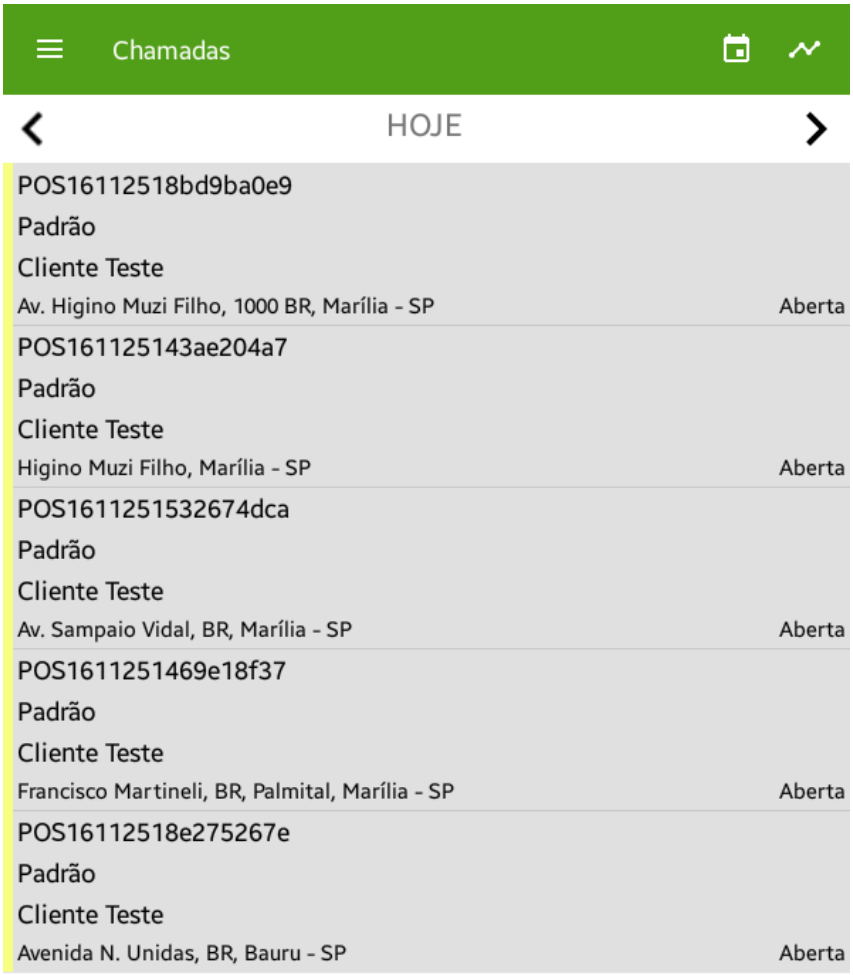

Figura 15 - Ordens ordenadas por distância

Fonte: Elaborado pelo autor, 2016.

Na figura 15, percebe-se que a ordenação das ordens foi alterada por distância, partindo do ponto utilizado pelo GPS (o teste foi realizado na UNIVEM, localizado em Marília-SP, Avenida Higino Muzi Filho).

Uma observação importante é que não necessariamente o usuário/agente precisa estar parado num local (por exemplo, ponto de origem). A cada vez que ele sincronizar as ordens de serviço, por exemplo: atendeu no ponto um e antes de ir para o ponto dois, ele sincroniza e percebe que há uma nova ordem com prioridade maior do que o ponto dois. Então, ele deve seguir para esta ordem de prioridade maior, ao invés de seguir ao ponto dois (supondo que ele estivesse atendendo por distâncias otimizadas, nesse caso, a prioridade de uma ordem vem primeiro do que a distância entre os pontos de atendimento).

5.4. Considerações Finais

Este capítulo foi essencial para mostrar a implementação e funcionamento do serviço desenvolvido num sistema corporativo. Foram feitos testes no Postman para verificar a confiabilidade do serviço, passando pontos iniciais, finais e pontos a serem percorridos na "vida real". Após testes iniciais realizados no Postman, o serviço foi implementado na arquitetura do sistema corporativo da eProdutiva. Além dos testes no Postman, foi realizado testes reais envolvendo agentes e ordens de serviço a serem cumpridas, passando localizações como endereço.

6. CONCLUSÃO

O objetivo deste trabalho foi o desenvolvimento de uma solução para o problema de alocação de tarefas com roteamento. Através desse objetivo, foi criado um serviço web (API) de roteirização, com foco em otimizar rotas e alocação de tarefas para agentes. Em relação a roteirização, a ideia do trabalho é organizar os pontos a serem percorridos por um agente, partindo de origem e passando por todos os pontos até o destino, buscando otimizar as rotas entre esses pontos.

O problema de alocação de tarefas com roteamento é um problema que fica cada vez mais difícil encontrar uma boa resolução ao aumentar exponencialmente o número de variáveis. Portanto, encontrar uma nova solução que possa resolver de maneira eficiente é muito importante.

O serviço foi bem sucedido em testes reais, pegando as latitude e longitude dos pontos e verificando distância entre eles, verificando prioridade caso houver. E no sistema corporativo, foi bem encaixado decidindo ao agente a melhor opção para economia de recursos como combustível para se deslocar entre os pontos de atendimento e consequentemente, no tempo gasto de um trajeto ao outro.

TRABALHOS FUTUROS

Diante da implementação realizada num sistema corporativo e sucesso nas aplicações de testes reais, o próximo passo é “aumentar” esse serviço, com objetivo de mostrar visualmente todo trajeto realizado, listando os pontos no mapa, mostrando a trajetória que deve ser realizada de acordo com algum critério específico como prioridade de ordem ou distância, de forma a mostrar ao usuário o caminho que ele irá percorrer.

Além disso, expandir o algoritmo para que resolva atribuição de multi agentes para multi tarefas.

REFERÊNCIAS

ALVES, Marcelo. **Resolvendo problemas com OptaPlanner.** Homepage. 2015. Disponível em: <http://www.devmedia.com.br/red-hat-resolvendo-problemas-de-planejamento-com-optaplanner-parte-1/31981 >. Acesso em 25 jun. 2016.

BUENO, Fabrício. **Métodos Heurísticos Teorias e Implementações**. 2009. Artigo. Instituto Federal de Santa Catarina, Campus Araranguá. Disponível em: < https://wiki.ifsc.edu.br/mediawiki/images/b/b7/Tutorial_métodos_heurísticos.pdf>. Acesso em: 10 jun. 2016.

CARVALHO, Rubens. **Problema da Mochila**, Universidade Estadual de Campinas Instituto de Matemática, Estatística e Computação Cientifica, 2015.

CHOPDE, Nitin R.; NICHAT, Mangesh K.; **Landmark Based Shortest Path Detection by Using A* and Haversine Formula**, International Journal of Innovative Research in Computer and Communication Engineering Vol. 1, Issue 2, April 2013. Disponível em: < https://www.researchgate.net/profile/Mangesh_Nichat/publication/282314348_Landmark_based_shortest_path_detection_by_using_A_Algorithm_and_Haversine_Formula/links/56389bb708ae4bde5021b0f5.pdf > Acesso em: 21 out. 2016.

CUNHA, C. B. **Aspectos práticos da aplicação de modelos de roteirização de veículos a problemas reais.** Revista Transportes da ANPET Associação Nacional de Pesquisa e Ensino em Transportes, v. 8, 2000.

DE SMET, Geoffrey, et. al.; **OptaPlanner User Guide**. Homepage. Disponível em: < https://docs.jboss.org/drools/release/6.1.0.Final/optaplanner-docs/html_single/#whatIsOptaPlanner >. Acesso em 24 jun. 2016.

EPRODUTIVA. Disponível em: <http://eprodutiva.com.br/>. Acesso em 20 nov. 2016.

FAN, DongKai; SHI, Ping. **Improvement of Dijkstra's Algorithm and its application in route planning.** Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2010.

FOWLER, Martin. **Microservices,** 2014. Disponível em: http://martinfowler.com/articles/microservices.html. Acesso em: 21 set. 2016.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação linear: modelos e algoritmos**. Elsevier Editora LTDA., 518 p, 2005.

HAKLAY, Mordechai; WEBER, Patrick. **OpenStreetMap: User-Generated Street Maps, IEEE Pervasive Computing**. Volume: 7, Issue: 4, Oct.-Dec. 2008. Disponível em: <http://ieeexplore.ieee.org/document/4653466/ >. Acesso em: 20 out. 2016.

HRISTAKEVA, M. ; SHRESTHA, D. **Different Approaches to Solve the 0/1 Knapsack Problem"**. Editora Campus, 2005. Acesso em: 13 out. 2016.

JACOBSON, Lee; KANBER, Burak. **Genetic Algorithms in Java Basics**. **Genetic Algorithms in Java Basics solve classical problems like the travelling salesman with GA**. Editora Apress, 2015.

JOHNSON, D. S.; MCGEOCH, L. A. **The traveling salesman problem: A case study in local optimization**, 1997.

KUHN, Harold W. **The Hungarian Method for the Assignment Problem**. ONR Logistics Project, Department of Mathematics, Princeton University, Princeton, United States, 1955.

LACERDA, Estefane G. M.; CARVALHO, André Carlos P. L. F**.; Introdução aos Algoritmos Genéticos**, 1999.

LACHTERMACHER, Gerson. **Pesquisa Operacional na Tomada de Decisões**. 2009.

LAPORTE, Gilbert et al**. Classical and modern heuristics for the vehicle routing problem.** International Transactions in Operational Research, v. 7, n. 4-5, p. 285-300, 2000.

LENSTRA, J. K. et al. **Complexity of vehicle routing and scheduling problems**. Networks, v. 11, 1981.

LINDEN, Ricardo. **Algoritmos Genéticos, uma importante ferramenta da Inteligência Computacional (2a Edição)**. Rio de Janeiro: Brasport, 2008.

LONGO, Humberto J. **Técnicas para Programação Inteira e Aplicações em Problemas de Roteamento de Veículos**. Tese para Doutorado. Pontifícia Universidade Católica do Rio de Janeiro, 2004. Disponível em: < http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=6029@1>. Acesso em: 11 out. 2016.

LUENBERGER, David G.; YE, Yinyu; **Linear and Nonlinear Programming.** Springer International Publishing Switzerland, v.228, n. 4, 555 p, 2016.

LUENBERGER, David G.; YE, Yinyu; **Linear and Nonlinear Programming, Fourth Edition.** Springer Science and Business Media, 2008.

LUQUE, L.; SILVA, R. R**. Algoritmos genéticos - conceitos e aplicação**. Java Magazine, 2010.

MACHADO, Fabricio C. et.al**.; Problema do Caixeiro Viajante com Coleta e Entrega de Objetos com Base Retangular**, 2015. Disponível em: < http://cdsid.org.br/sbpo2015/wp-content/uploads/2015/08/142759.pdf>. Acesso em: 06 set. 2016.

MARTELLO, S.; TOTH, P. **Knapsack Problems: Algorithms and Computer Implementations.** John Wiley & Sons, Chichester. 1990.

MAURI, Geraldo Regis. **Uma nova abordagem para o problema de roteirização e programação de veículos.** 2006. Monografia apresentada para o Exame de Qualificação do Curso de Doutorado em Computação Aplicada, INPE. Acesso em: 18 out. 2016.

NAMIOT, Dmitry; SNEPS-SNEPPE, Manfred. **On micro-services architecture**, 2014.

NEWMAN, Sam. **Building Microservices**, O'Reilly Media, Inc. 2015.

NUNES, Rosângela da Silva; GUIMARÃES, Norton C.; CARVALHO, Cedric L.; **Planejamento de Grade de Horário em uma Universidade Brasileira envolvendo algoritmos genéticos**, 2013.

OPTAPLANNER. Homepage. 2015. Disponível em: <www.optaplanner.org>. Acesso em: 10 out. 2016.

PEREIRA, Dilson L. **Heurísticas e Algoritmo Exato para o Problema de Roteamento de Veículos com Coleta e Entrega Simultâneas.** Dissertação de Pós-Graduação. Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, 2010. Disponível em: < https://www.dcc.ufmg.br/pos/cursos/defesas/1209M.PDF>. Acesso em: 17 out. 2016.

POSTMAN. **Postman is how people build and test API's.** Disponível em: < https://www.getpostman.com>. Acesso em: 20 nov. 2016.

PRESTES, Álvaro Nunes. **Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante**, 2006.

ROCHA, Ítalo Mendonça**. Uma abordagem otimizada para o problema de alocação de equipes e escalonamento de tarefas para a obtenção de cronogramas eficientes**. 2011. Dissertação de Mestrado. Universidade Estadual do Ceará. Disponível em: <http://www.uece.br/macc/index.php/arquivos/doc_download/196-uma-abordagem-otimizada-para-o-problema-de-alocacao-de-equipes-e-escalonamento-de-tarefas-para-a-obt>. Acesso em: 10 out. 2016.

SAMPAIO, R. M. **Estudo e Implementação de Algoritmos de Roteamento**. 1998.

SANTANA, Marcelo N. P. **Alocação de Tarefas Paralelas Comunicantes em Ambientes Distribuídos Heterogêneos**. Dissertação de Mestrado. Universidade de Brasília. Instituto de Ciências Exatas, Departamento de Ciência da Computação, 2006. Disponível em: < http://repositorio.unb.br/handle/10482/2175>. Acesso em: 11 out. 2016.

SOARES, Henrique C. A. **Um estudo sobre o Problema de Alocação**. 2011. Monografia. Universidade Federal de São Paulo. Disponível em: < http://www.ft.unicamp.br/docentes/meira/publicacoes/2011henrique.pdf>. Acesso em: 07 mai. 2016.

VANDERBEI, Robert J. **Linear Programming: Foundations and Extensions, Second Edition. Department** of Operations Research and Financial Engineering, Princeton University, Princeton, 2001. Disponível em: < https://support.dce.felk.cvut.cz/pub/hanzalek/_private/ref/Vanderbei_Linear_Programming.pdf>. Acesso em: 13 out. 2016.

VANDERBEI, Robert J. **Linear Programming: Foundations and Extensions, Third Edition.** Springer, 2007.

VIDEIRA, Aline S. **Uma proposta de resolução para problemas de roteirização de veículos baseada no algoritmo genético.** 2011**.** Monografia. Acesso em: 12 set. 2016.

ZHANG R.; ZHOU S. **The Application of The Improved Hybrid Ant Colony Algorithm in Vehicle Routing Optimization Problem.** 2nd International Conference on Future Computer and Communication, IEEE. 2010. Disponível em: < http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5497706>. Acesso em: 30 mar. 2016.

ANEXO - AUTORIZAÇÃO DE USO DAS MARCAS E DO SOFTWARE

persys www.persys.com.br

AUTORIZAÇÃO DE USO DAS MARCAS

PERSYS e eProdutiva

A PERSYS Projetos de Inovação Tecnológica EIRELI, inscrita pelo CNPJ 13.601.126/0001-63, representante exclusiva da marca do sistema de gestão eProdutiva, autoriza o Leonardo Yoshiharu Miyazawa, RG 41.119.118-4, a utilizar as marcas PERSYS, eProdutiva e as informações contidas no sistema web e móvel em seu o Trabalho de Conclusão de Curso (Bacharelado) de Ciência da Computação do Centro Universitário Eurípides de Marília (Fundação de Ensino Fundação de Ensino Eurípides Soares da Rocha de Marília).

Marília, 25 de novembro de 2016

Giulianna Marega Marques
Sócio - Administrador

Página 1 de 1

PERSYS Projetos de Inovação Tecnológica
Marília/SP (14) 3434-0744

Figura 16 - Autorização de uso das marcas e do software