UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

HELDER ROCHA DA SILVA

BOATROUTE: USO DO ALGORITMO DIJKSTRA A FIM DE DETERMINAR O MELHOR CAMINHO PARA PERCURSOS NÁUTICOS NA REGIÃO DE SANTA CATARINA

CRICIÚMA
2017

HELDER ROCHA DA SILVA

BOATROUTE: USO DO ALGORITMO DIJKSTRA A FIM DE DETERMINAR O MELHOR CAMINHO PARA PERCURSOS NÁUTICOS NA REGIÃO DE SANTA CATARINA

Trabalho de Conclusão de Curso, apresentado para a obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador(a): Profª. MSc. Christine Vieira

CRICIÚMA

2017

HELDER ROCHA DA SILVA

BOATROUTE: USO DO ALGORITMO DIJKSTRA A FIM DE DETERMINAR O MELHOR CAMINHO PARA PERCURSOS NÁUTICOS NA REGIÃO DE SANTA CATARINA

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de bacharel, no Curso de curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Teoria dos grafos.

Criciúma, 27 de novembro de 2017.

BANCA EXAMINADORA

Prof Kristian Madeira - Dr - UNESC

Prof Fabricio Giordani - Esp -UNESC

Prof Christine Vieira - MSc -UNESC -Orientador

A Rafaela Rocha que sempre está ao meu lado até nas tempestades mais difíceis para vencendo sempre com senso de humor.

AGRADECIMENTOS

A minha orientadora Chistine Vieira, pelo suporte no pouco tempo que lhe coube, pelas correções, incentivos e exigências. Tive a felicidade de escolher a professora certa.

Ao professor e amigo William Bertan da Silva , que sempre nos incentiva a ser perseverantes com nosso objetivos, além de todo conhecimento que pode me passar.

A colega e amiga Caroline Salib Canto, que em toda a trajetória estudantil esteve junto comigo sendo minha cúmplice no sucesso e aprendizado com seu apoio e dedicação .

A amiga e escritora Deise Duarte, que me ajudou a revisar todos os textos nos últimos dias e também pelas palavras de apoio.

Aos meus amigos, que direta ou indiretamente fizeram parte da minha formação me dando apoio e incentivo para alcançar esta etapa na vida.

“Você meio que começa a pensar que tudo é possível se você tiver nervos suficiente.”

J.K. ROWLIN

RESUMO

O uso da tecnologia se difunde diariamente para a automatização de diversas atividades, e para área náutica ela tem muito a agregar. O projeto propõe a combinação do algoritmo de Dijkstra em conjunto de conceitos aplicados para que se atinja uma navegação mais rápida a fim de oferecer um aplicativo que possa auxiliar no processo de busca por percursos mais ágeis. Para a definição de conceitos e aplicação do aplicativo foi realizado levantamento bibliográfico e uma busca por ferramentas livres de desenvolvimento a fim de oferecer um protótipo que estimasse os melhores percursos de uma rota. O projeto trouxe resultados positivos gerando o protótipo BoatRout sendo possível encontrar melhores percursos e estimar uma média de tempo para as rotas.

Palavras-chave: Algoritmo de Dijkstra. Área náutica**.** Dispositivos móveis. Melhor percurso de navegação.

ABSTRACT

The use of technology diffuses daily for the automation of various activities, in the nautical area it has much to add. The project proposes the combination of the Dijkstra algorithm and applied concepts to achieve faster navigation in order to offer an application that can aid in the process of searching for more agile routes. For the definition of concepts and application of the application was carried out bibliographical survey and a search for development in order to offer a prototype that estimated the best routes of a route. The project brought positive results generating the BoatRout prototype being possible to find better routes estimating the average time for the routes.

Palavras-chave: Dijkstra Algorithm. Nautical area**.** Mobile devices. Best navigation course.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ADT	Android Development Tools
AVD	Android Virtual Device
API	Application Programming Interface
GPS	Global Positioning System
IDE	Integrated Development Environment
SDK	Software Development Kit
SMS	Short Message Service
XML	Xtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O uso da tecnologia se difunde diariamente em prol da simplificação de problemas de determinadas áreas, agilizando ou automatizando serviços. Como em quaisquer outras atividades dentro da cultura de navegação o uso de inovações tecnológicas vem para agregar. Muitos softwares estão disponíveis para uso dentro da área naval, porém a grande maioria necessita do auxílio de equipamentos para melhor funcionamento.

Entre a aparelhagem necessária, podemos encontrar microcontroladores, sinais dos satélites do *Global Positioning System* (GPS), dentre outros. Entre as que ajudam navegadores de pequenas embarcações, destacam-se computadores de bordo que tem como função informar o posicionamento do barco em mapas digitais, direção, e outras informações úteis para auxiliar em uma navegação segura e inteligente. O uso de muitos equipamentos em pequenas embarcações, como veleiros, por exemplo, torna-se difícil. Mesmo que estes barcos sejam de um tamanho maior, o investimento de equipagem torna-se alto (BORTOT, 2009).

Visando buscar soluções relacionadas ao espaço físico, o desenvolvimento de aplicativos *mobile* se difunde gradativamente. O uso de *smartphones* está mais presente no cotidiano, e a quantidade de aplicativos para os dispositivos, com diversas funcionalidades e objetivos, cresce no mesmo ritmo. Aplicativos voltados para a navegação marítima em aparelhos móveis estão sendo lançados e abrangem desde o planejamento de rotas à orientações de usuários comuns. Mesmo o utilizador podendo determinar qual aplicativo irá usar em seu celular, com foco em geonavegação, em muitos casos acaba não utilizando um software de auxílio, pois em grande parte os aplicativos disponíveis não são gratuitos para download ao público alvo (COSTA, 2015).

Com o uso da tecnologia GPS integrada no *smartphone*, foram desenvolvendo-se aplicativos úteis para descoberta de caminhos e rotas. Alguns programas que auxiliam os navegadores utilizando inovação são: Garmin Mobile XT, Maplink Destinator, Route 66, BeeLineGPS. Eles tendem a auxiliar embarcações náuticas na estimativa de menores rotas, mostrando o

posicionamento do barco em um mapa, entre outras funções que possam beneficiar os navegadores (MORIMOTO, 2009).

Sabendo da existência de aplicações já desenvolvidas, pode-se encontrar as mesmas disponíveis a partir de tarifas pagas ou disponibilizadas em versões *trials*, gratuitas apenas para testes de curto prazo. Dentre os valores dos investimentos, os preços iniciais para aplicativos náuticos na *PlayStore* alternam-se de R$ 11,00 (onze reais) a R$ 150 (cento e cinquenta reais). No entanto, muitos dos usuários de embarcações de pequeno porte são considerados hobbista, a maioria opta por não comprar e consumir este tipo de softwares devido ao investimento necessário.

Perante a situação apresentada, referindo-se ao custo para a acessibilidade dos softwares de apoio em navegações marítimas, o objetivo desta pesquisa é oferecer uma nova solução gratuita, criando um aplicativo a partir de tecnologias livres e usando o algoritmo de Dijkstra para o cálculo da melhor rota a ser seguida pela embarcação, sendo este um dos algoritmos mais utilizados para determinação de percursos mínimos. O aplicativo também irá utilizar o GPS integrado à um dispositivo que contenha o sistema operacional *Android*. O aplicativo proposto tende a sanar o planejamento de rotas, e buscar o percurso mais rápido, caso não encontre a rota mais ágil oferecer a rota de melhor ou menor distância.

O protótipo proposto prioriza a busca por rotas mais rápidas ou as de menor distância, levando em consideração informações como o vento, correnteza dentre outras variáveis necessárias para a otimização da navegação. Além disso utilizando os recursos oferecidos pelo GPS para que nos forneça uma distância para um percurso. O protótipo a ser construído será disponibilizado na forma de código-aberto, gratuitamente e com o instalador numa conta do *GitHub*, com o intuito de difundir softwares móveis para apoio à navegação marítima no estado de Santa Catarina, buscando mapear pontos náuticos recomendados para tráfego marítimo.

1.1 OBJETIVO GERAL

Desenvolver um protótipo para dispositivos móveis que realize o cálculo da melhor ou menor rota em um percurso náutico em Santa Catarina utilizando o algoritmo de Dijkstra.

1.2 OBJETIVOS ESPECÍFICOS

Para o desenvolvimento deste projeto de pesquisa foram determinados os seguintes objetivos:

a) pesquisar o algoritmo de caminho mínimo Dijkstra;
b) compreender cálculos para estimativas em rotas náuticas;
c) pesquisar o funcionamento de um GPS;
d) compreender funcionalidade do Java para a plataforma *Android*;
e) criar um protótipo para aplicativos móveis para o cálculo dos percursos náuticos utilizando Dijkstra, em dispositivos com sistema operacional *Android*.

1.3 JUSTIFICATIVA

Antes de viajar pode-se utilizar aparelhos com GPS, buscando calcular a distância e tempo estimado entre ponto de saída e destino. Para a área náutica não é diferente, antes de uma navegação é primordial a verificação de distâncias entre dois pontos, como também informações climáticas referentes ao mar, dentre outras. A fim de realizar estas estimativas muitos softwares foram sendo lançados para a área de navegação, entretanto a grande maioria dos softwares disponíveis são pagos ou disponibilizados em versões triais, para um uso de demonstração em um curto espaço de tempo. Também existem outras aplicações que além de o valor para a adesão, o usuário precisa de alguns equipamentos para trabalhar em conjunto. Para solucionar ambos os problemas, a criação de um aplicativo gratuito voltado para dispositivos móveis tende a ser a melhor saída, pois com o mesmo podemos utilizar o GPS do próprio dispositivo, quando disponível, e a loja de aplicativos da plataforma para a divulgação do software. Moreira (2013) afirma que o sistema operacional *Android*

é o mais utilizado atualmente, pois é compatível com diversos dispositivos mobile e acaba não exigindo um hardware tão robusto. O sistema operacional *Android* também disponibiliza para os desenvolvedores alguns benefícios, como por exemplo a abstração do hardware para o desenvolvedor, sendo possível o acesso aos recursos de forma transparente, rápida e segura.

Ele acaba por intermediar comandos das ferramentas integradas ao hardware, onde os mesmos possam tirar total proveito que um aparelho portátil pode oferecer.

Em busca de um percurso para um determinado destino, muitas vezes encontramos mais de um caminho. Para a solução deste tipo de problema é muito comum a utilização do uso de algoritmos baseados na Teoria dos Grafos. De acordo com Silva (2011), a teoria dos Grafos vem sendo vista como uma das áreas mais importantes dentro da matemática discreta, sendo utilizada em construções de modelos que geram maior visibilidade de problemas de um ângulo proporcionalmente melhor.

Segundo Campos (2007), dentro da teoria dos grafos existe um vasto estudo procurando encontrar algoritmos para determinar um menor caminho entre dois pontos. Os problemas de Caminho Mínimo compreendem a determinação do percurso de menor tamanho (distância, tempo ou um custo qualquer) entre dois nós.

Entre vários caminhos, aquele que possui o menor “peso” é chamado de caminho mínimo. Através das somas destes pesos podemos calcular uma estatística para o tempo de uma viagem, a distância percorrida.

O algoritmo mais famoso por sua eficiência em determinar o caminho mínimo entre dois pontos é o algoritmo de Dijkstra. Silva (2011) aponta que o algoritmo já foi utilizado para sanar problemas de trânsitos em cidades, rotas de metrô dentre outras facilitações. Pela efetividade conhecida referindo-se ao algoritmo de Dijkstra, será estudado sua aplicação para assim colaborar nesta pesquisa.

Utilizando técnicas de algoritmos de caminho mínimo e os recursos que o sistema operacional *Android* pode oferecer, sendo este um intermediário

para ser possível a utilização de recursos do hardware, pode-se criar uma aplicação portátil com pouco investimento para o usuário final.

A fins de validação do estudo proposto, será reduzida à áreas populares de navegação da região de Santa Catarina, mapeando algumas das rotas disponíveis.

1.4 ESTRUTURA DO TRABALHO

Os temas abordados pelo trabalho foram distribuídos em cinco capítulos, sendo o primeiro composto pela introdução, objetivo principal, objetivos específicos, justificativa e a própria estrutura do trabalho, aqui relatada. Quanto aos demais capítulos, os assuntos abordados foram:

a) Teoria dos grafos: este capítulo tem como objetivo esclarecer assuntos relacionados a teoria dos grafos como definições básicas de grafos e técnicas relacionadas à simplificações dos mesmos;

b) O algoritmo de Dijkstra: este capítulo aborda as definições do algoritmo de Dijkstra e como o mesmo se comporta quando aplicado em um grafo;

c) Global Position System: este capítulo informa a história desta tecnologia abordando a intenção inicial da mesma até a evolução e aplicação para *smartphones*;

d) Cartas náuticas: este capítulo demonstra os dados que as cartas náuticas podem oferecer quando necessário para encontro das rotas a serem navegadas;

e) Como são estimados os recursos náuticos: O capítulo explica como são estimados os percursos, técnicas para uma navegação mais ágil e também como pode-se estimar uma média da velocidade do barco em movimento;

f) Trabalhos correlatos: neste capítulo são informados trabalhos que utilizam variações do algoritmo de Dijkstra e aplicações destas para solucionar diversos problemas;

g) Implementação do protótipo BoatRoute: este capítulo demonstra toda a metodologia utilizada para o desenvolvimento do

protótipo, além de demonstrar os resultados obtidos com o trabalho desenvolvido;

h) Conclusão: aqui será concluído o trabalho, demonstrando as experiências e ideias obtidas com este projeto, além de ser levantado possíveis trabalhos futuros relacionado a área de aplicação do presente trabalho.

2 TEORIA DOS GRAFOS

Teoria dos grafos é uma área um ramo da matemática que estuda objetos de um determinado conjunto. É uma ferramenta muito importante no ramo da matemática, usada a fim de modelar uma variedade de problemas, nas mais diversas áreas, tais como: Genética, Inteligência Artificial, Química, Linguística, Pesquisa Operacional, Geografia, Engenharia Elétrica, entre outras (NICOLETTI; HRUSCHKA JUNIOR, 2013).

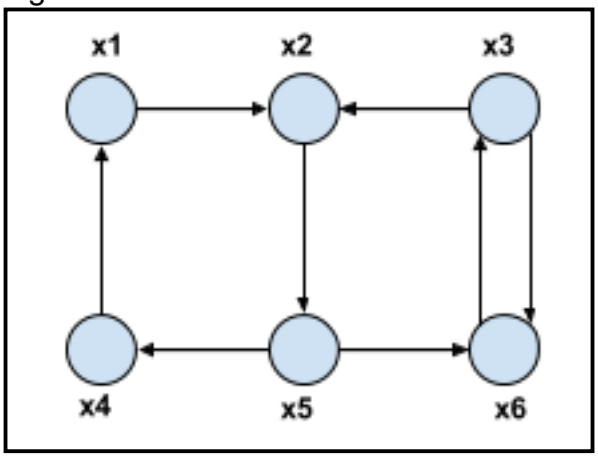
Grafo é uma simples noção, genérica e logicamente intuitiva, que é usada representando a ideia de alguma espécie de relação entre os “objetos”. Graficamente, é representado por uma imagem que possui nós ou vértices, representando os objetos, unidos por traços chamados de arestas, caracterizando a relação imaginada (ALOISE; CRUZ, 2001).

2.1 PRINCIPAIS CONCEITOS

A analogia abordada por Pereira (2014) explica que um grafo *G= (V,E)* é um sistema formado por um conjunto *V* de elementos chamados vértices, pontos ou nodos, e um conjunto *E* de pares não ordenados de vértices chamados linhas ou arestas. Em algumas literaturas pode ser apresentado o termo *V(G)* e *E(G)* para explicitar que *V* e *E* são respectivamente, os conjuntos de vértices e de linhas do grafo *G* (NICOLETTI; HRUSCHKA JUNIOR, 2013).

Um caminho é uma cadeia na qual todos os arcos possuem a mesma orientação. A sequência de vértices (x_1, x_2, x_5, x_6, x_3) é um exemplo de caminho no grafo representado na figura 1.

Figura 1 - Grafo orientado.

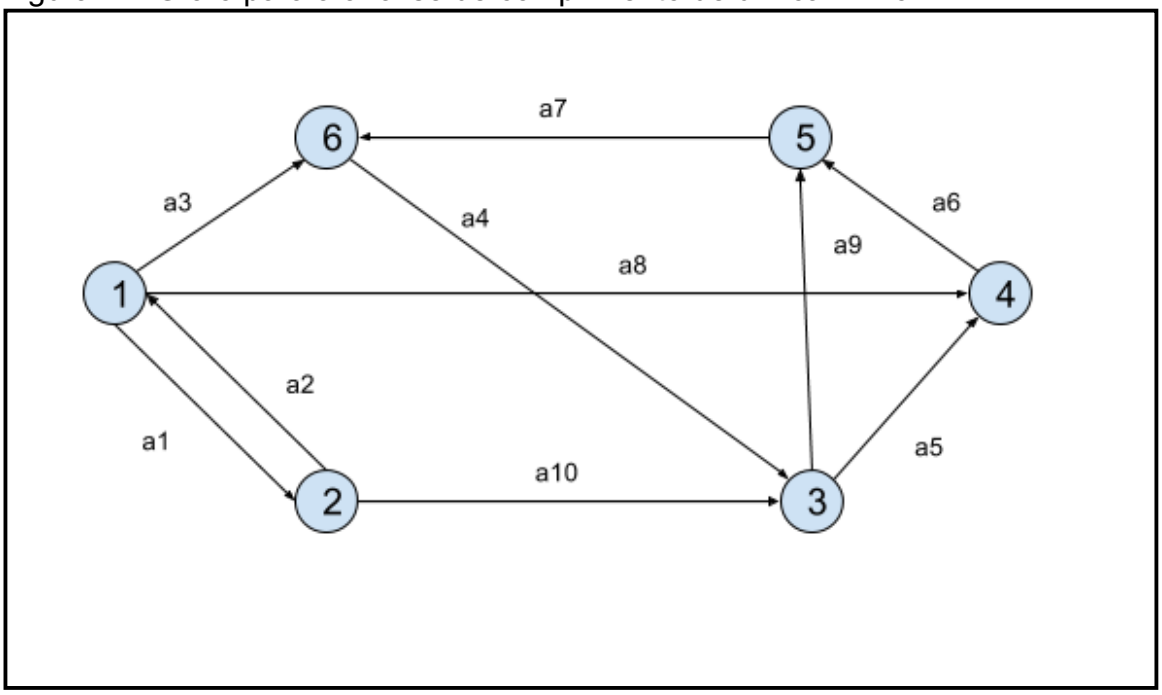


Fonte: Do Autor.

O comprimento de um caminho é dado pelo número de vértices que o mesmo contém. Se um arco for usado mais de uma vez, ele é contado cada vez que é usado.

Na figura 9, o comprimento do caminho do nó 6 para o nó 5 (6, a4, 3, a9, 5) é 2 e o comprimento do caminho do nó 4 para o nó 3 (4, a6, 5, a7, 6, a4, 3) é 3, conforme o grafo na figura 2.

Figura 2 – Grafo para a análise de comprimento de um caminho



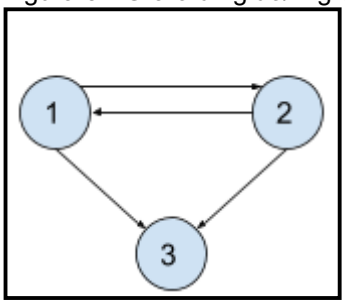
Fonte: Do autor.

2.1.1 Grafo dirigido e não dirigido

Grafos não dirigidos e grafos dirigidos (dígrafos) são representados habitualmente por meio de diagramas em que os vértices são ilustrados por pequenos círculos e as arestas são linhas. Também podendo possuir uma seta que indica a sua direção ou orientação, ou seja, qual o vértice de origem e qual o vértice de destino.

São considerados dígrafos ou grafos dirigidos, aqueles que em sua estrutura possuem uma seta que indica a sua direção ou orientação, ou seja, qual o vértice de origem e qual o vértice de destino, ou seja, o respectivo par ordenado. Na figura 3 apresenta-se a representação do diagrama do grafo dirigido D= (V, A), onde V representa os vértices do grafo e A as arestas; V ={1,2,3}, A = {(1,2), (2,1), (1,3), (2,3)}. Pode-se observar que as arestas são um par ordenado, onde o primeiro elemento é o vértice de origem e o segundo elemento deste par ordenado, é o vértice de destino.

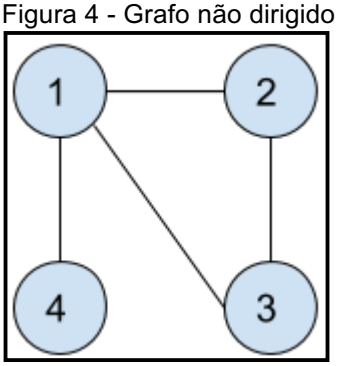
Figura 3 - Grafo dirigido/Dígrafo



Fonte: Do autor.

Um grafo não dirigido, não é completamente padronizado, não sendo necessário à existência de indicação de direção na estrutura do grafo. Neste caso as arestas não são pares ordenados (PEREIRA, 2014). Na figura 4 tem-se a representação do diagrama do grafo não dirigido G= (V, E) onde V'= {1,2,3,4}, E={[1,2], [2,3], [3,1],}, E= {[1,2], [2,3], [3,1], [1,4]}.

Figura 4 - Grafo não dirigido.



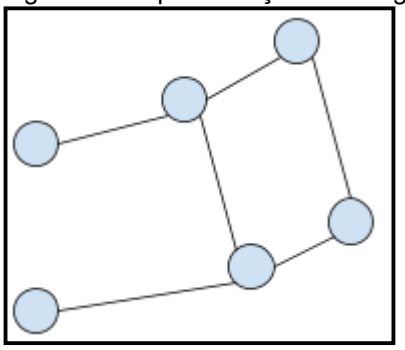
Fonte: Do autor.

2.1.2 Grafo simples

Se duas ou mais arestas do grafo tem os mesmos vértices-extremidade, estas são chamadas de arestas paralelas (NETTO; JURKIEWICZ, 2011).

Um laço em um grafo é uma aresta com extremidades n – n para algum vértice n, isto significa que a extremidade da aresta é sempre o mesmo vértice.Um grafo simples é aquele não possui laços e nem arestas paralelas, como está representado na figura 5 (NETTO; JURKIEWICZ, 2011).

Figura 5 - Representação de um grafo simples.

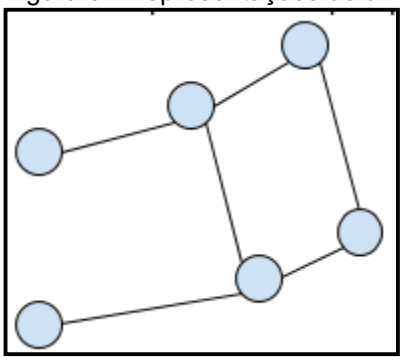


Fonte: Do autor.

2.1.3 Grafo conexo

Um grafo é dito conexo quando existe um caminho de um vértice do grafo para qualquer outro vértice do grafo, conforme ilustra a figura 6. Neste grafo é possível encontrar um caminho de um vértice para qualquer outro, por isso ele é dito como sendo um grafo conexo.

Figura 6 - Representações de um grafo conexo.

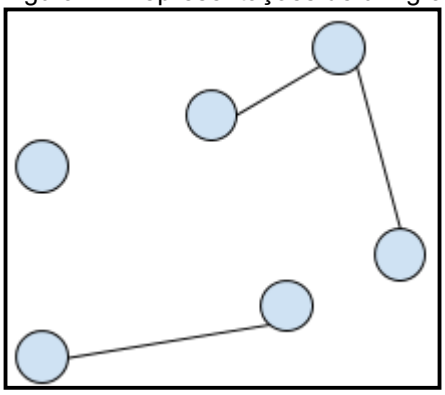


Fonte: Do autor.

2.1.4 Grafo desconexo

Netto e Jurkiewicz (2011), dizem que para este tipo de grafo, nem todos os vértices tem caminho para os demais vértices do grafo, como pode ser observar na figura 7.

Figura 7 - Representações de um grafo desconexo.

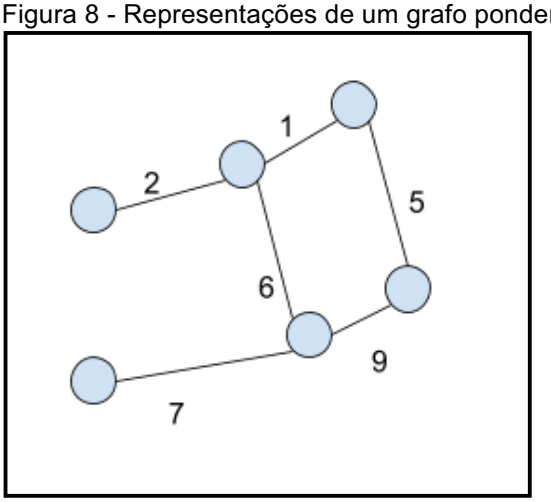


Fonte: Do autor.

2.1.5 Grafo ponderado

Silva (2009) explica que um grafo ponderado possui pesos associados a cada uma de suas arestas, conforme está ilustrado na figura 8.

Figura 8 - Representações de um grafo ponderado.



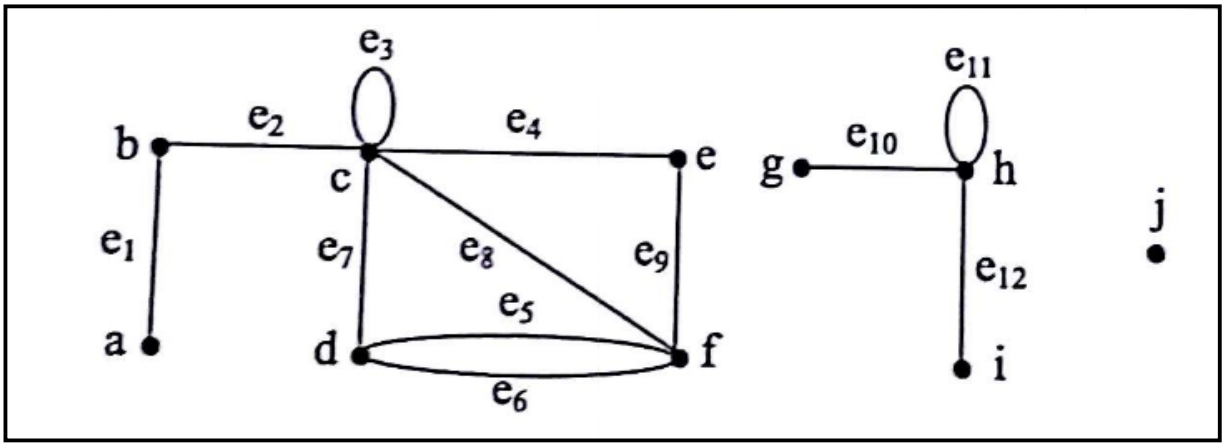
Fonte: Do autor.

2.1.6 Relações de adjacência

Em um grafo simples, costuma-se dizer que vértices vizinhos são vértices adjacentes.

Dois vértices que estão unidos por uma aresta são chamados de adjacentes ou vizinhos, como por exemplo, os vértices a e b da figura 9. Duas arestas distintas ei e ej são adjacentes se tem um vértice em comum, como pode ser visto nas arestas e_1 e e_2 na figura (NICOLETTI; HRUSCHKA JUNIOR, 2013).

Figura 9 - Demonstração de grafos para comparação dos vértices



Fonte: Niccoletti e Hruschka Junior (2003, p. 32).

2.1.7 Percursos

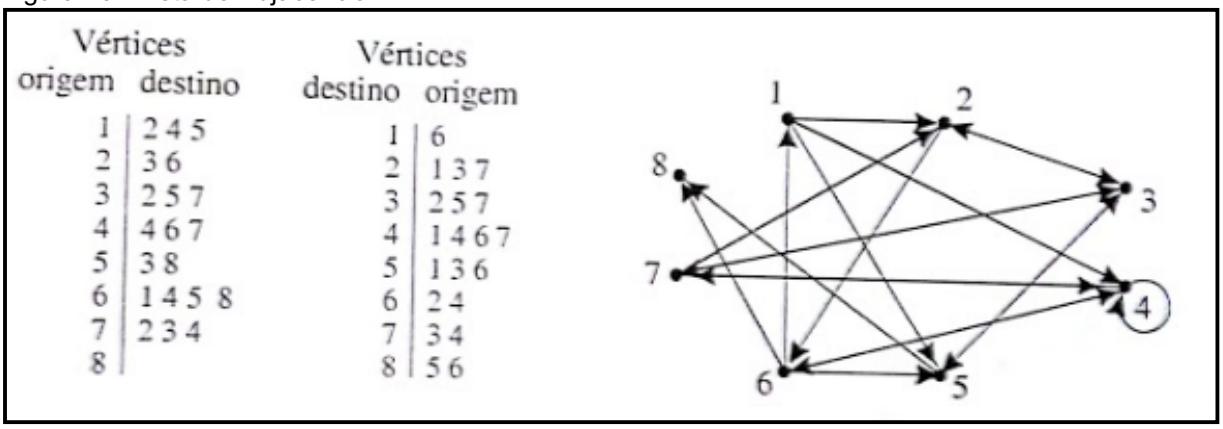
Sequências de arestas sucessivamente adjacentes são denominadas de percurso, cada uma tendo uma extremidade adjacente à anterior e a outra a subsequente, com exceção da primeira e da última (NETTO; JURKIEWICZ, 2011).

2.1.8 Representação de grafos

Para esquematizar a estrutura de um grafo busca-se a utilização de técnicas de armazenamentos a fim de que seja de fácil entendimento à estrutura de um grafo.

Lista de adjacência (ou dicionário) é uma das formas de representação, construída como um conjunto de listas de vértices, cada lista sendo formada por um vértice destino e origem e pelo conjunto de vértices, o mesmo tem ligação através de uma aresta como representado na figura 10 (NETTO; JURKIEWICZ, 2011).

Figura 10 - Lista de Adjacência

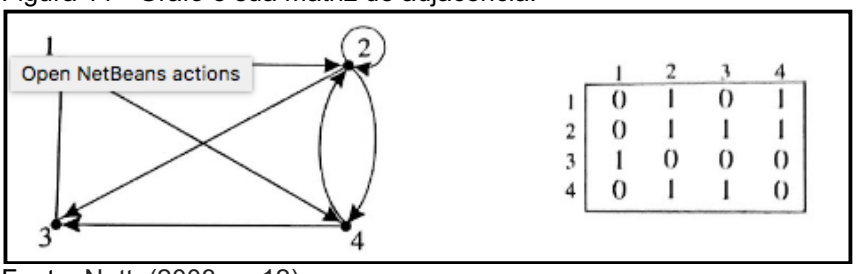


Fonte: Netto (2003, p. 11)

Segundo Netto (2003), matriz de adjacência é uma das diversas formas de representação matricial, trate-se de uma matriz de ordem **n**, onde **n** é a quantidade de vértices que o grafo possui e na qual associa-se cada linha e cada coluna da matriz. Para popular essa matriz é verificada linha por linha se o vértice da linha atual possui alguma ligação com o vértice da coluna atual. Como por exemplo, no vértice 1 representado na matriz pela linha 1 e coluna 1, conforme listra a figura 11. Não há nenhuma aresta que ligue o vértice 1 com ele

mesmo, logo este campo é preenchido com *0*, porém seguindo a mesma linha 1 com a coluna 2, identifica-se uma ligação do vértice 1 com o vértice 2, sendo assim o campo da matriz é preenchido com o valor *1*.

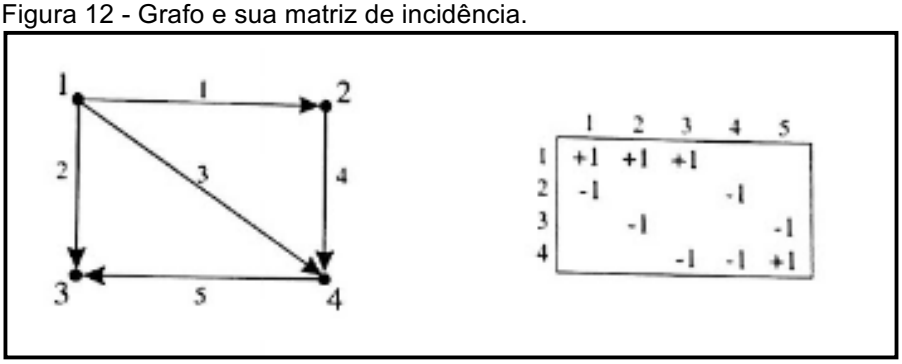
Figura 11 - Grafo e sua matriz de adjacência.



Fonte: Netto(2003, p. 12)

Matriz de incidência é uma matriz de vértice por arestas, onde as arestas representam as colunas e os vértices as linhas. Como por exemplo a matriz da figura 12. Para preencher a matriz verifica-se em cada coluna se há ligação com o vértice da linha, caso exista, este campo na matriz é preenchido com os valores +1 e -1. O valor +1 é atribuído ao vértice de origem e o valor -1 é atribuído ao vértice de destino (NETTO, 2003).

Figura 12 - Grafo e sua matriz de incidência.



Fonte: Netto(2003, p. 13)

2.2 CAMINHO MÍNIMO

O problema de menor caminho em um grafo consiste em determinar o menor caminho entre um vértice inicial e um vértice final.

Netto e Jurkiewicz (2011), dizem que na teoria dos grafos, o problema do caminho mínimo consiste na minimização para percorrer um grafo entre dois vértices.

Este problema pode ser modelado através de um grafo, aonde os pontos de entrada e saída são representados pelos nodos, a cada um deles sendo associado um nome. As arestas correspondem ao caminho entre os pontos, sendo que o caminho deve ser percorrido apenas pela direção das flechas de cada aresta, adicionalmente aparece um peso associado a cada aresta que corresponde ao setor do caminho indicado possui um peso que corresponde. O custo do caminho é dado pela soma de cada aresta percorrida e pelo cálculo destes valores, existem alguns algoritmo que podem ser aplicados.

3 O ALGORITMO DE DIJKSTRA

O algoritmo de Dijkstra é um dos mais conhecidos para resolução do problema de menor caminho entre dois pontos de um grafo.

3.1 EDSGAR WYBE DIJKSTRA

Edsgar Wybe Dijkstra foi um cientista da computação nascido na Holanda em 11 de maio de 1930. Filho de mãe matemática e um pai químico, o mesmo graduou-se em Matemática e Teoria da Física em 1956 na Universidade de *Leinden* e mais tarde, em 1959, recebeu seu título de PhD pela Universidade de Amsterdam com sua tese intitulada, '*Communication with an Automatic Computer*', que em português ficaria, '*A Comunicação com Computadores Automáticos*', na qual se dedicava a descrever a montagem da linguagem projetada para o primeiro computador comercial desenvolvido na Holanda, o X1. Edgar posteriormente veio a ser conhecido por sua contribuição à Teoria dos Grafos com o algoritmo de Dijkstra (KRZYSZTOF, 2002, tradução nossa).

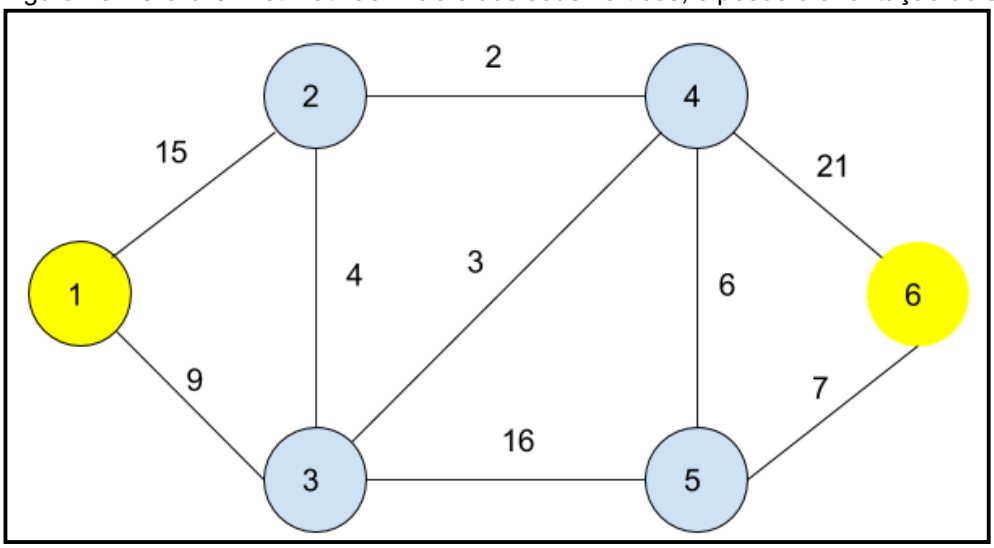
Dijkstra em 1959, após estudos, propôs um algoritmo para resolver dois problemas de conexões em grafos: o menor caminho e árvore geradora mínima com fonte única. O problema dos caminhos mínimos é um clássico tema em otimização combinatória. Sendo um grafo G com pesos não negativos em suas arestas (grafo ponderado) e um vértice fonte s, encontrar caminhos (de pesos) mínimos de s a todos os demais vértices .Dado um grafo com arestas ponderadas, o problema da árvore geradora mínima baseia-se em encontrar uma árvore em que todos os vértices se conectem (direta ou indiretamente) uns aos outros. A estrutura deve possuir o menor peso possível, onde o peso é dado pela soma dos valores das arestas escolhidas (mínima) (SILVA, 2009).

O algoritmo de Dijkstra é um algoritmo que se fundamenta em localizar o caminho mínimo entre um vértice inicial e um vértice escolhido do grafo como ponto final de uma trajetória, não necessariamente que este caminho de custo mínimo passará por todos os vértices (nós) do grafo (NETTO; JUTKIEWICZ, 2011).

3.2 ETAPAS PARA UTILIZAÇÃO DO ALGORITMO DE DIJKSTRA

Considere o grafo ponderado G=(V, A) da figura 13, sendo *1* o vértice raiz de G. Com a finalidade de ilustração, será calculado o caminho de custo mínimo do vértice *1* ao vértice *6* usando o algoritmo de Dijkstra.

Figura 13 - Grafo G: Estimativas iniciais dos seus vértices, e pesos e orientação de suas arestas



Fonte: Do autor.

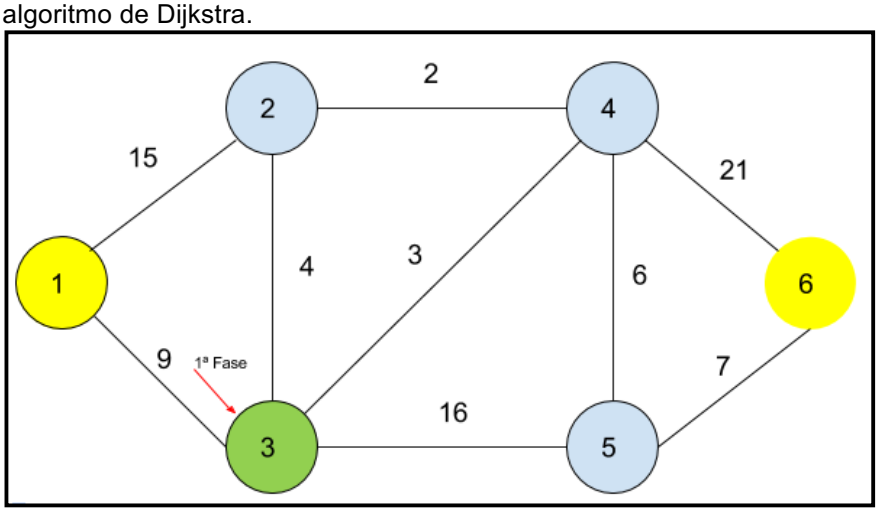
Inicialmente, todos os vértices têm estimativa de custo infinito, exceto o de número 1, vértice escolhido como raiz, que tem custo zero. O vértice S (vértice 1) encontra-se aberto, pois é o vértice raiz. O conjunto S de vértices fechados está vazio.

3.2.1 Primeira etapa

Inicialmente parte-se do vértice 1 com o intuito de percorrer um caminho até o vértice 6, onde a soma dos trajetos deve corresponder ao menor valor. Para melhor didática de entendimento foram sugeridos valores em cada caminho, estes valores por sua vez simbolizam as dificuldades das rotas marítimas encontradas. Os vértices adjacentes ao vértice 1 são os vértices 2 e 3.

Logo se percebe que existem os caminhos com valor 15, ligado ao vértice 2 e o caminho 9, ligado ao vértice 3. A regra é orientar pelo caminho de menor valor, sendo assim o caminho percorrido nessa primeira etapa é do vértice 1 ao 3, simbolizado pelar cor verde como indica a figura 14.

Figura 14 - Grafo G: Ilustração da primeira etapa percorrida no grafo com utilização do algoritmo de Dijkstra.

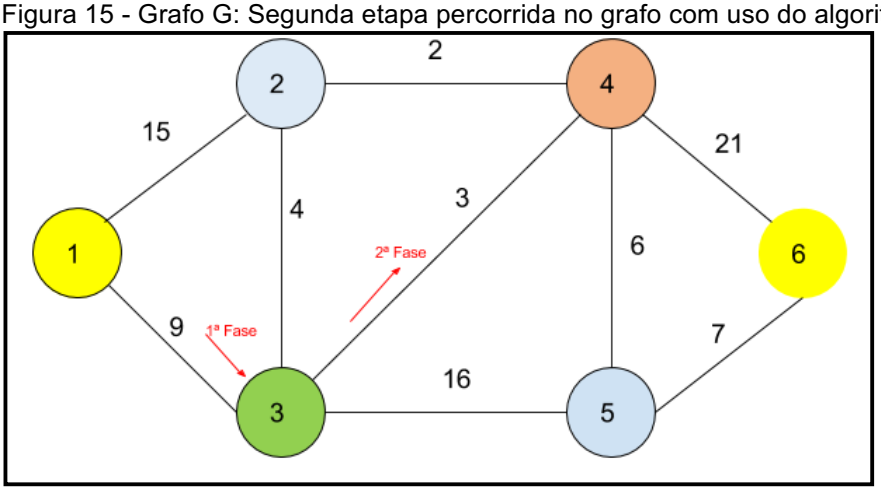


Fonte: Do autor.

3.2.2 Segunda etapa

Do vértice 3 escolhido como trajetória de menor valor, a escolha agora é pelos caminhos adjacentes ao vértice 3, que são os vértices 4 ou 5. Por sua vez o trajeto de menor valor liga o vértice 3 ao 4, simbolizado pela cor laranja (figura 15).

Figura 15 - Grafo G: Segunda etapa percorrida no grafo com uso do algoritmo de Dijkstra.

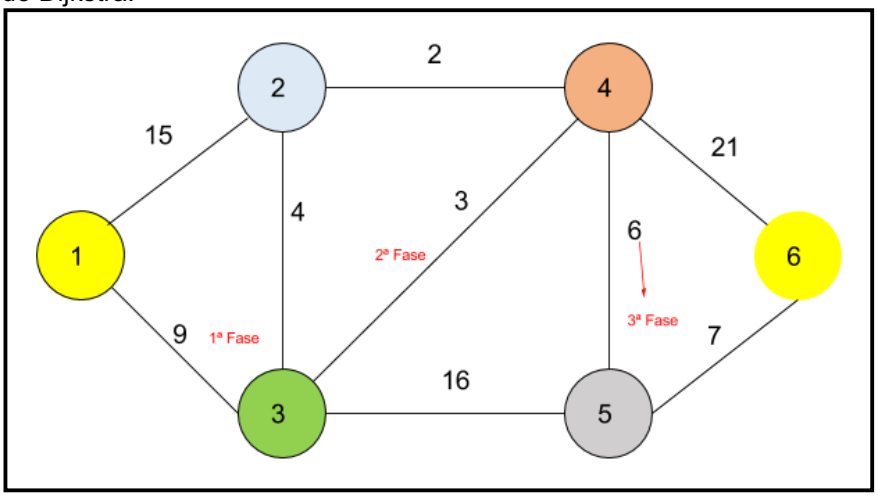


Fonte: Do autor.

3.2.3 Terceira etapa

Do vértice 4 há duas possibilidades: a trajetória que liga até ao vértice 6 ou ao vértice 5, esse último com menor valor, simbolizado pela cor cinza conforme figura 16.

Figura 16 - Grafo G: Ilustração da terceira etapa percorrida no grafo com utilização do algoritmo de Dijkstra.

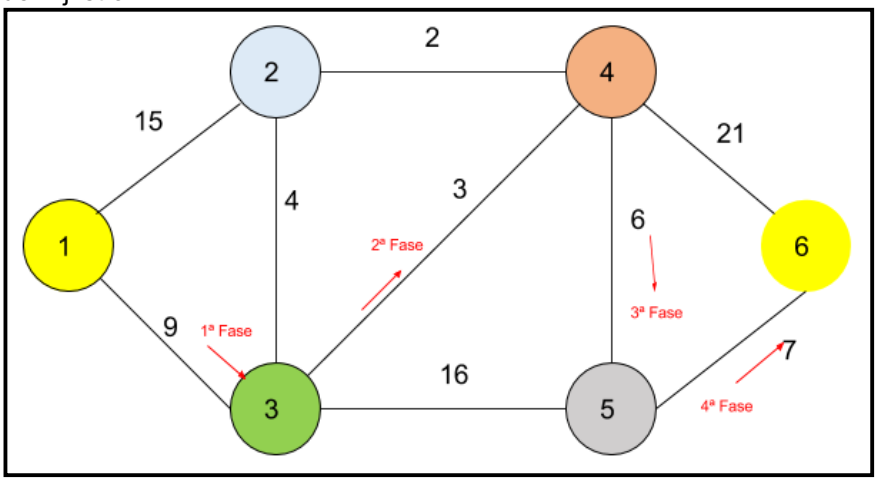


Fonte: Do autor.

3.2.4 Quarta etapa

Para finalizar o caminho e assim chegar à rota prevista (vértice 6) toma-se como trajetória o caminho que liga o vértice 5 ao 6 de acordo com a figura 17.

Figura 17 - Grafo G: Ilustração da quarta etapa percorrida no grafo com utilização do algoritmo de Dijkstra.



Fonte: Do autor.

4 GLOBAL POSITION SYSTEM

O uso da métodos manuais para navegação vem sendo substituído de maneira gradativa por tecnologias mais modernas, em especial o Global Position System (GPS), por estas serem capazes de atribuir maior precisão à localização espacial em qualquer parte do globo imprimindo maior confiabilidade à navegação.

É possível compreender a utilização de mecanismos modernos, como o GPS, para se atribuir a localização exata da embarcação alinhada à possibilidade de inserir estes pontos em cartas náuticas, que em descrição simples, consistem em mapas aquaviários. Tem-se nessa união de fatores, o nascimento de um sistema capaz de imprimir cada vez mais segurança à navegação moderna.

A popularização do uso de sistemas GPS na sociedade moderna transmite, em certa medida, uma relativa impressão de simplicidade quanto ao seu funcionamento. De acordo com Faria (2016), pode-se dizer inclusive, que seu advento representou revolução semelhante à observada quando do descobrimento das Américas, basta ver seu enorme impacto na navegação geo-espacial, quer seja no campo civil, quer seja na utilização para fins militares.

O sistema teve seu início pelo Departamento de Defesa Americano na década de 60 do século passado, por meio de um projeto militar conhecido como NAVSTAR. Entretanto o sistema que atualmente conhecemos só foi considerado completo em 1995, ou seja, desenvolvido ao longo de trinta e cinco anos, com um custo final estimado em mais de dez bilhões de dólares americanos (HOWELL, 2013, tradução nossa).

Formado por uma rede composta por 32 satélites geo-estacionários, conforme Morimoto (2009), o sistema permanece em constante atualização, visto que o GPS originalmente era formado por apenas 24 satélites. Atualmente, conforme pesquisa realizada no sítio Geoduc (2015), o número de satélites em órbita no sistema GPS permanece em 32, mas outros sistemas semelhantes ao americano, como o sistema Russo conhecido por Globalnaya Navigatsionnaya Sputnikovaya (GLONASS) ou Sistema de Navegação Global por Satélite estão

em operação ou em vias de implantação, o que tem auxiliado para o desenvolvimento contínuo da tecnologia em sistemas de localização espacial.

O Departamento de Defesa Americano criou a chamada Interferência Proposital ou Disponibilidade Seletiva, para o uso civil do sistema GPS, aumentando a margem de erro da localização com vistas à inibição de possíveis usos não pacíficos, como por organizações terroristas. Esta, interferência foi aumentada pelos atrasos causados pelos elétrons livres presentes na ionosfera (FARIA, 2016). Ainda segundo Faria (2016), essa interferência é inexistente para o uso militar, oferecendo ao sistema uma margem de precisão a depender do receptor utilizado.

Para tamanha precisão na obtenção de posicionamento geográfico no globo terrestre, o sistema utiliza-se da rede de satélites citada anteriormente. Conforme Morimoto (2009), seu funcionamento consiste na emissão, por parte da rede de satélites em órbita, de um sinal de rádio contendo um pacote de informações a ser processada pelo receptor em terra. Através da utilização das informações resultantes de no mínimo três satélites, o sistema processa, através de uma triangulação, o local exato da localização em solo. À medida que o receptor se desloca, o tempo para recepção do sinal vai se alterando, o que faz com que o sistema atualize a informação da localização. Ainda é importante ressaltar que a rede geo-estacionária foi criada de modo que, em qualquer lugar do planeta, sempre haja no mínimo três satélites visíveis aos equipamentos em solo, o que atribui ao sistema a capacidade de localização geográfica em qualquer ponto na Terra.

Grandemente utilizado para fins civis, atualmente os GPS são uma mistura de coordenadas de localização com mapas em três dimensões e seu uso vai além do simples posicionamento geográfico, sendo utilizados para calcular rotas de trânsitos, por exemplo, agregadas a serviços como velocidade em vias de trafego e rotas alternativas.

Juntamente ao desenvolvimento contínuo da tecnologia de posicionamento global, a tecnologia moderna dos hardwares permitiu a miniaturização dos receptores de GPS, permitindo que os mesmos fossem operados em smartphones, tabletes, e até mesmo em relógios de pulso,

promovendo além da democratização de seu uso o barateamento generalizado dos custos de aquisição desses receptores. De qualquer forma, a utilização de todas as tecnologias até aqui apresentadas, em especial as cartas náuticas e os GPS, ainda estão limitadas à localização espacial e ao traçado de rotas de navegação.

5 CARTAS NÁUTICAS

A navegação via georeferência, que compõe a utilização de pontos de referência marcantes e alinhamento estelar, relembra às mais antigas civilizações que utilizavam este mecanismo em suas rotas, representando o início da busca, pelos navegadores, de forma que visavam a garantir os melhores e mais seguros caminhos entre dois pontos.

A empreitada ibérica em direção ao novo mundo, após muitas tentativas frustradas de encontrar uma rota alternativa às índias, resultou aos navegadores futuros as chamadas cartas náuticas. Na época, e mesmo em dias atuais, estas cartas representaram uma evolução sem paralelo se comparadas aos métodos primitivos de geolocalização. São instrumentos revolucionários que ainda hoje constituem-se na linha mestra de referência náutica em termos de rotas de navegação (BESSONE; RIBEIRO; GONÇALVES, 2016).

A cartas náuticas são o resultado do levantamento de regiões oceânicas, mares, baías, rios, canais, lagos, ou qualquer outra massa d'água navegável e que tem como objetivo central a garantia de rotas seguras para a navegação. Em sua grande maioria são elaboradas na projeção de Mercator, uma exibição cilíndrica elaborada por geógrafos, cartógrafos e matemáticos. Apresentam as eventualidades terrestres e submarinas, fornecendo informações tais como profundidade e perigos à navegação (bancos, pedras submersas, cascos sobrados ou qualquer outra barreira). Buscam ainda informar dados da natureza do fundo marinho como fundeadouros e áreas de fundeio além de

apontarem a instrumentos de auxílio à navegação (faróis, faroletes, boias, balizas, luzes de alinhamento, radiofaróis). O desenvolvimento constante das cartas permitiu ainda a inclusão de altitudes e pontos importantes aos navegantes, linhas de costa e de contorno das ilhas, elementos de marés, correntes e magnetismo, além de outras indicações necessárias à segurança da navegação (FERNANDES, 2012).

O grande problema da navegação existe no fato que, durante esta, perde-se a noção do que se encontra a sua volta bem como o que pode existir abaixo da linha d´água. Observa-se, portanto, que o problema da segurança da navegação ainda é persistente nos dias atuais e a utilização das cartas náuticas tem por finalidade a mitigação dos riscos envolvidos nas viagens náuticas. A garantia da localização se dá através da utilização de coordenadas de latitude e longitude sendo as primeiras informadas nas laterais e as últimas nos campos superiores e inferiores da carta. A utilização cruzada de ambas as informações, permite ao navegador verificar sua localização (BRASIL, 2017).

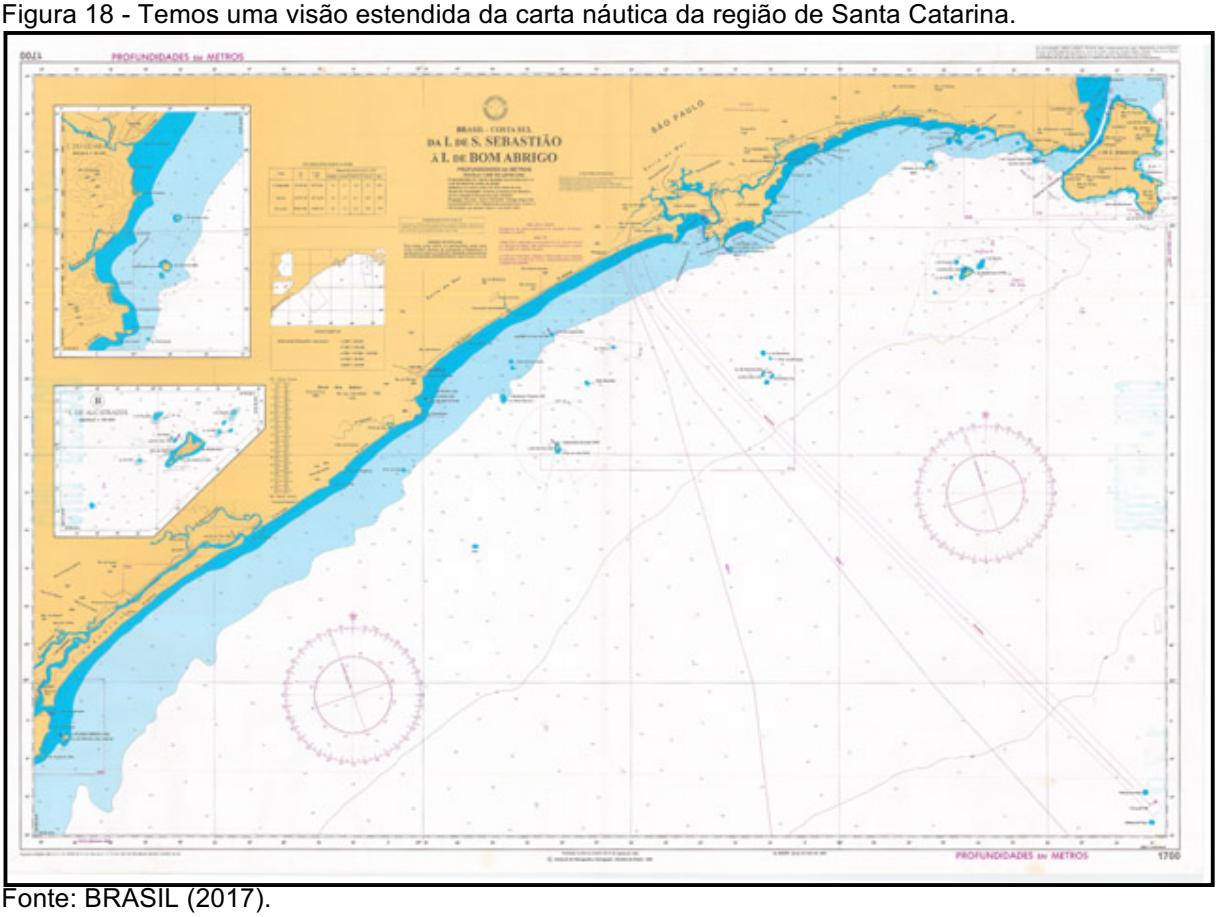
5.1 ROTAS NÁUTICAS DE SANTA CATARINA

Muitos navegadores têm como costume criarem páginas na internet para informar e detalhar regiões de navegação, para assim compartilhar experiência e dicas para uma vagem mais segura.

Ribeiro (2007) narra em seu diário de viagem disponível online, que a costa marítima de Santa Catarina é muito rica, nela podemos encontrar todos os tipos de desafios para uma grande navegação. Pequenos trechos, obstáculos com rochas, faróis, reservas naturais dentre outras.

Na figura 18 fornecida pela Brasil (2017) tem-se um exemplo de um mapa náutico de Santa Catarina. Esta carta mostra de forma ampla a costa marítima do estado. Neste tipo de mapa podemos ter uma previsão de profundidade, altitudes, sinais de navegação dentre outros detalhes.

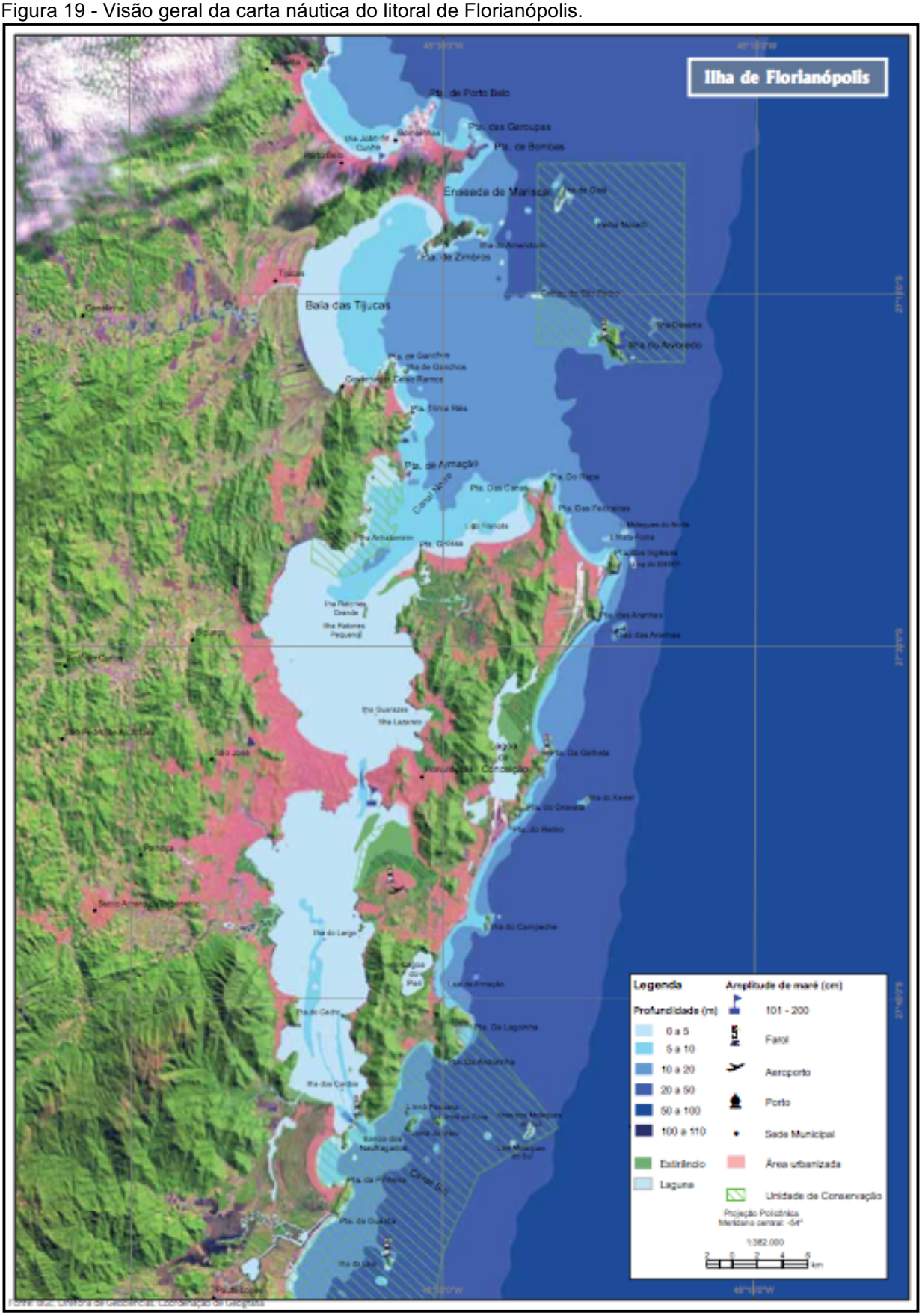
Figura 18 - Temos uma visão estendida da carta náutica da região de Santa Catarina.



Fonte: BRASIL (2017).

O IBGE (2011) fornece uma documentação com as principais cartas náuticas do Brasil, focando na região de Santa Catarina. O mesmo oferece um mapa da ilha de Florianópolis, em que se podem ver detalhes úteis para uma navegação na região. A figura 19 apresenta uma visão macro da carta náutica da região de Florianópolis e na figura 20 encontra-se um quadro qual detalha-se pontos importantes do mapa.

Figura 19 - Visão geral da carta náutica do litoral de Florianópolis.



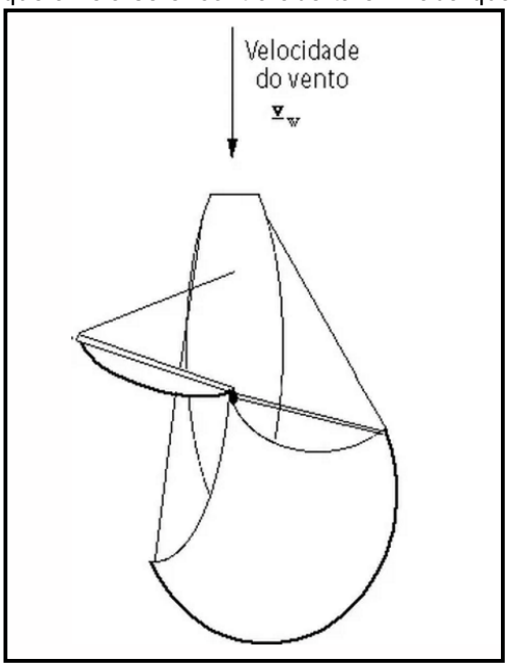
Fonte: IBGE (2011, p.17).

6 COMO SÃO ESTIMADOS OS RECURSOS NÁUTICOS

Para que seja estimado uma rota rápida os navegadores planejam a partir de variáveis climáticas que podem aparecer em seus percursos. Dentre os itens dos quais os navegadores necessitam ter maior conhecimento, temos correnteza e vento sendo os principais variáveis.

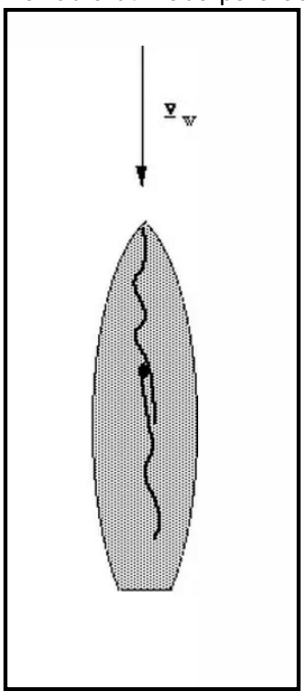
Referindo-se ao vento, encontramos duas teorias para navegação sendo estas, o veleiro pode navegar com o vento a favor da vela, conforme ilustrado na figura 21, onde, o vento sopra nas velas e desloca o barco. Nesta situação, o vento é mais rápido do que o barco, de forma que o ar é desacelerado pelas velas assim empurrando utilizando a força gerada para movimentar o barco. Entretanto um barco que navega a favor do vento sempre possui uma menor velocidade, mas nesta situação é mais confortável para o navegador pois o barco ganha maior estabilidade, não ocorrendo grande oscilação de movimentos. Mas não é o mais interessante na navegação quando pretende-se chegar em menor tempo. Também existe a possibilidade de navegar contra o vento, os barcos a vela podem navegar contra ao vento, tendo uma aproximação de 40° em relação ao vento, para conseguir atingir esta manobra a vela tende a ficar mais fechada quase em linha reta com a ponta do barco, conforme ilustrado na figura 22 (WOLFE, 2002, tradução nossa).

Figura 20- Representação da vista de cima de um veleiro navegando a favor do vento, note-se que a vela se encontra aberta e inflada quando atingida por uma grande quantidade de vento.



Fonte: Wolfe (2002).

Figura 21 - Ilustração de um veleiro com a vela aproximado da reta da ponta do barco, manobra utilizada para obter velocidade e navegar contra o vento.



Fonte: Wolfe (2002).

Levando em consideração a correnteza da água, a navegação não a favor da correnteza é possível, entretanto o barco terá que passar por manobras

mais difíceis para adquirir a força contracorrente. Em resumo o ideal para uma navegação mais rápida seria ao encontro da corrente a favor sendo e contra o vento para que a força de empuxo da vela seja ainda maior (WOLFE, 2002).

6.1 CALCULO DE VELOCIDADE ALTERNADA DO BARCO

O movimento de um barco em relação às águas chama-se *movimento relativo*. Buscando saber a velocidade de um barco pode-se utilizar a relação vetorial da Física.

Para este temos os seguintes conceitos a adotado por Nicolau (2013), para a retirada dos valores das variáveis, sendo:

O movimento das águas que arrastam o barco em relação às margens é o *movimento de arrastamento.*

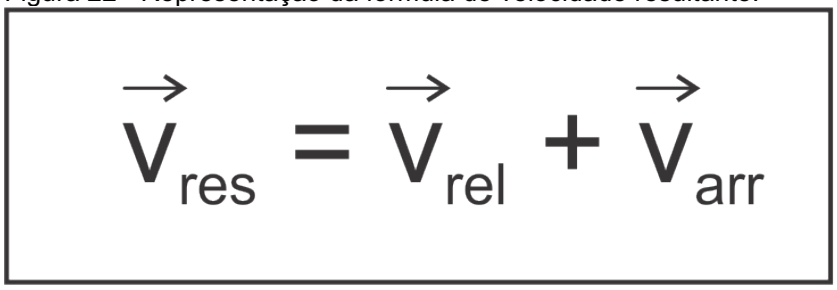
O movimento do barco em relação às margens, isto é, em relação à Terra, é o movimento resultante.

A velocidade do barco em relação às águas é a velocidade relativa (v_{rel}).

A velocidade das águas, isto é, a velocidade da correnteza é a *velocidade de arrastamento* (v_{arr}).

A velocidade do barco em relação às margens é a *velocidade resultante* (v_{res}).Tem-se a relação vetorial, representado na figura 22.

Figura 22 - Representação da formula de velocidade resultante.



Fonte: Do autor.

Portanto, a velocidade do movimento resultante é a soma vetorial das velocidades dos movimentos relativo e de arrastamento.

7 ANDROID

O Android é um sistema operacional desenvolvido para a utilização em dispositivos móveis. Sua estrutura foi desenvolvida baseando-se em um *kernel* do Linux, o que o torna *Open Source*, possui seu código aberto, tornando possível o acesso do código aos desenvolvedores para que estes possam realizar colaborações, correções e desenvolvimento de novos aplicativos (JACKSON, 2011, tradução nossa).

Toda a estrutura não foi desenvolvida por uma única empresa e sim por um grupo chamado *Open Handset Alliance (OHA)*, onde estão envolvidas organizações mundialmente conhecidas como Dell, Intel, Motorola, Samsung e outros, além da Google que lidera o grupo, a plataforma de desenvolvimento para aplicativos móveis Android possui uma interface rica, Global Positioning System (GPS), várias aplicações já instaladas, um poderoso ambiente de desenvolvimento, além de utilizar a linguagem de programação Java (LECHETA, 2013).

Desenvolvedores Android podem desenvolver e distribuir seus aplicativos através da loja virtual da *Google Play*, também conhecida como *Play Store*, loja de aplicativos da plataforma Android, nela é possível disponibilizar aplicações gratuitas ou pagas para todo o mundo, visto que a loja atende todos os usuários de dispositivos com sistema Android (DEVELOPER, 2015, tradução nossa).

7.1 TECNOLOGIAS ENVOLVIDAS NO DESENVOLVIMENTO ANDROID

As aplicações Android desenvolvidas tem em foco dispositivos móveis utilizando o Android SDK, ferramenta que está disponível sem custos e taxas para os desenvolvedores. A linguagem de desenvolvimento pode ser escolhida conforme a preferencia de cada um, partindo da ideia que existe mais de uma linguagem compatível com o Android, sendo elas: Java, C/C++, Python, Ruby, Hypertext Markup Language (HTML) e Javascript (GOK; KHANNA, 2013, tradução nossa).

Uma das possíveis interfaces para o desenvolvimento de uma aplicação em Java para Android é o Ambiente Integrado de Desenvolvimento, do inglês *Integrated Development Environment (IDE) Eclipse*. Que é uma plataforma de código aberto para desenvolvimento, sendo possível utilizar outras linguagens de programação nesta IDE Eclipse, bastando adicionar alguns *plug-ins* (LUCKOW; MELO, 2010).

Facilitando o desenvolvimento Android o Google criou no ambiente de desenvolvimento *Eclipse* um plug-in conhecido como *Android Development Tools* (ADT) que também ajuda nos testes e na compilação do projeto. Através dele é possível utilizar o emulador direto da IDE Eclipse e todos os seus recursos, como debug que executa passo a passo o código, controlar o emulador, ter acesso aos logs, simular uma chamada ou uma mensagem chamada de *Short Message Service (SMS)*, além de enviar e visualizar arquivos através do emulador (LECHETA, 2013)

Uma das mais recentes IDEs utilizada para o desenvolvimento Android é o Android Studio, esta ferramenta foi adotada como o ambiente de desenvolvimento oficial do Android. Que dispõe de todos os recursos necessários, como o editor de código inteligente, capaz de refratorar e analisar os códigos apontando os possíveis erros e sugerindo o que deve ser alterado, permite as simulações do aplicativo em diversos tamanhos de telas e tipos de dispositivos, como *Tablets, Smartphones, Android Wear, Android TV e Google Glass*. Pode ser feito o download da ferramenta de forma gratuita (DEVELOPER, 2015).

8 TRABALHOS CORRELATOS

Durante a pesquisa bibliográfica foram encontrados artigos e trabalhos que abordam temas semelhantes aos abordados no presente projeto.

8.1 UTILIZAÇÃO DO ALGORITMO DE DIJKSTRA PARA CÁLCULO DE ROTAS NO TRABALHO PÚBLICO DO MUNICÍPIO DE CRICIÚMA/SC

O autor buscou apresentar de maneira objetiva, a implementação de um software para cálculo das melhores rotas de trabalho do transporte público de Criciúma/SC.

Para a metodologia, foi constituído um levantamento bibliográfico sobre a aplicação do algoritmo de Dijkstra para solução de melhores caminhos. Com os conhecimentos obtidos através das pesquisas, foram aplicadas programações do Dijkstra na linguagem PowerScript, no ambiente Powerbuilder 9.0 com integração ao banco de dados, no ambiente Adaptive Server Anywhere 9.0. Implementando um grafo dinâmico das rotas de Criciúma, onde o posicionamento dos vértices e arestas era lido do banco de dados. O ambiente proporciona uma grande interação visual ao usuário, sendo intuitivo e de fácil utilização. Os locais escolhidos para aplicação do algoritmo no do auxílio da resolução do problema de trânsito, foram as rotas de trabalho público em Criciúma/SC.

Os autores relatam que com os resultados obtidos através dos estudos, os mesmos puderam desenvolver um software com a junção do algoritmo de Dijkstra utilizando um problema real de trânsito na cidade de Criciúma/SC. O objetivo geral foi alcançado com sucesso, finalizando com um simples programa e de fácil utilização, onde o usuário pode calcular as melhores rotas dos caminhos que escolheu para o transporte público do município de Criciúma/SC (GUIZZO; ZANETTE; SCARPATO, 2017).

8.2 INVESTIGAÇÃO SOBRE A OTIMIZAÇÃO DO ALGORITMO DE DIJKSTRA E SUA APLICAÇÃO

Estrutura de dados, que possui um significado muito importante para melhorar a eficiência da resolução do algoritmo de caminho mais curto.

A metodologia seguiu através de pesquisas históricas, matemáticas e aplicações do algoritmo, através de testes com linguagens de programação, como o C++ por exemplo.

A conclusão foi a aplicação de uma melhoria do algoritmo Dijkstra, para obter o caminho mais curto de um nó para outro. Segundo os autores, o trabalho ainda precisa evoluir, mesmo assim, aprimorou-se muito o espaço de tempo em que o algoritmo levava para percorrer um grafo (RK; REDDY; SHAMA, 2015, traducao nossa)

8.3 UTILIZAÇÃO DE UM ALGORITMO DE CAMINHO MÍNIMO NO PROCESSO DE RECOLHIMENTO DO PALHIÇO DA CANA-DE- AÇÚCAR

Nesta dissertação, o autor relata grandes problemas das queimadas dos canaviais, e a consequente geração de palhiço por este feito.

Segundo Silva (2009), o trabalho teve como objetivo propor aplicações de técnicas matemáticas de otimização a fim de facilitar o planejamento do recolhimento do palhiço da cana-de-açúcar, que pode ser aproveitado na geração de energia.

Para a metodologia Silva (2009), conta que foram estudados melhores formas de carregamento dos fardos de palhiço para facilitar o transporte, assim diminuir custos e desgastes de maquinários. Para isto, foi sugerido o uso de técnicas de agricultura de precisão a fim de mapear o palhiço enfardado, desta forma podendo-se definir uma rota para recolher os fardos no campo e transportá-los para o centro de processamento. Para 2 determinação da rota, foi proposto o uso do algoritmo de menor caminho da teoria de grafos, utilizando uma variação do algoritmo de Dijkstra.

Em conclusão Silva (2009) diz que o trabalho pode-se evidenciar que as técnicas matemáticas de otimização tem sido uma excelente ferramenta para auxílio a gestão do setor sucroalcooleiro, contribuindo para comprovar que o recolhimento do palhiço, é tecnicamente viável e pode ser utilizado para geração de energia como componente relevante na matriz energética brasileira.

Silva (2009), afirma que o objetivo proposto teve sucesso. O método utilizado permitiu a definição de uma rota mínima para recolhimento dos fardos de palhiço no talhão. O método proposto é de fácil implementação computacional, e pode ser utilizado como ferramenta nos processos de gestão de empresas, definindo o caminho mínimo para recolhimento dos fardos de palhiço no campo, mapeados através de técnicas da Agricultura de Precisão, o que auxiliaria na redução de custos.

8.4 TRÊS ALGORITMOS DE CAMINHOS MÍNIMOS EM REDES RODOVIÁRIAS REAIS: ESTRUTURAS DE DADOS E PROCEDIMENTOS

Este artigo analisa uma aplicação teórica a fim de resolver problemas de congestionamento de filas para o uso de transportes públicos rodoviários e resume os três algoritmos de busca de uma caminho mínimo, dentre eles o algoritmo de Dijkstra, com uma de suas variações que visualiza o algoritmo de crescimento do gráfico implementado com duas filas, também outra variação onde é implementado com filas aproximadas, e por fim uma terceira variação onde é implementado o algoritmo Dijkstra com filas duplas.

No artigo também foram demonstradas as estruturas de dados e os procedimentos relacionados aos algoritmos. Segundo o autor este artigo deve ser particularmente útil para pesquisadores e profissionais de transportes, pesquisas operacionais e ciências de gestão.

A metodologia conforme abordada pelo autor se baseou em pesquisar detalhadamente cada uma das variações do algoritmo e analisar de forma lógica e teórica como seria a aplicação de cada destas alternâncias dentro dos ambientes de transportes.

Em conclusão, pode-se evidenciar que para cada cenário de transporte rodoviário, se aplica uma variação diferente dos algoritmos a fim de sanar as necessidades em particular, para organizar o fluxo de filas nas distintas rodoviárias (ZHAN, 1997, tradução nossa).

8.5 METODOLOGIA BASEADA NO ALGORITMO DE DIJKSTRA PARA ROTEIRIZAÇÃO DE PONTOS TURÍSTICOS EM OURO PRETO

Qual é o caminho mais curto até um destino? Esta pergunta, apesar de sua simplicidade, possui uma complexidade relativamente grande. Isto é colocado, pois existem diferentes interpretações desta: especificação de menores rotas entre dois pontos, cálculo de centros de rotas, determinação de rotas com restrições estudo de velocidades para se calcular rotas dentre outras.

Diante deste contexto foi abordado a determinação de rotas entre pontos de um grafo de distâncias na cidade de Ouro Preto. Para determinar o caminho mínimo entre os pontos foi preciso a utilização do Algoritmo de Dijkstra.

A metodologia utilizada foi o estudo de caso, criando situações e detalhando a dificuldade de alguns pontos em uma rota, assim obtendo embasamento para aplicar estes conhecimentos no desenvolvimento de um protótipo.

Concluindo, a pesquisa foi possível implementar o protótipo no ambiente de programação Matlab com o qual pode-se ter resultados de boas rotas turísticas que evitassem transtornos nos percursos. O Algoritmo de Dijkstra mostrou-se eficiente para a determinação destes percursos (SILVA, 2010).

8.6 UTILIZAÇÃO DO ALGORITMO DE DIJKSTRA PARA RESOLVER O PROBLEMA DO CAMINHO MÍNIMO EM MAPAS CONSTRUÍDOS COM O FORMATO SCALABLE VECTOR GRAPHICS

Neste projeto de conclusão de curso, o autor mostra a implementação de um protótipo para auxiliar a localização do menor caminho entre determinados pontos representados por um grafo em imagens SVG. Utilizando o algoritmo de Dijkstra e implementado este recurso com a linguagem de programação Javascript e a API DOM.

Para metodologia do trabalho, foi necessário o entendimento dos algoritmos que auxiliam na busca pelo menor caminho entre pontos modelados através de um grafo.

Um destes algoritmos foi implementado utilizando a linguagem computacional Javascript e Document Object Model (DOM) sobre mapas construídos no formato Scalable Vector Graphics (SVG), além de utilizar arquivos eXtensible Markup Language (XML) para o armazenamento dos dados. As próximas seções apresentarão os conceitos levantados neste estudo

O trabalho obteve como resultado o protótipo de um sistema que auxilia na localização do menor caminho, utilizando a teoria dos grafos e o algoritmo de Dijkstra, sendo implementados na linguagem Javascript além de contar com a API DOM sobre imagens SVG (COELHO, 2004).

9 IMPLEMENTAÇÃO DO PROTÓTIPO BOATROUTE

A aplicação de algoritmo de Dijkstra é utilizada há muito tempo para descobertas de caminhos mínimos, sendo utilizado em conjunto de algumas variações que levam em consideração regras para sanar diferentes situações.

Na implementação do protótipo BoatRoute, o algoritmo de Dijkstra tem como função a busca do trajeto de navegação mais rápido, ou seja, o melhor caminho pois descarta percursos que contenham algum obstáculo. Caso não

exista nenhuma via respeitando as regras definidas para uma navegação mais veloz o protótipo então utiliza o algoritmo para procurar o percurso de menor distância.

Para uma jornada de navegação são pesquisadas rotas que desviem de grandes obstáculos e dentro desses trechos são procurados percursos em que o trajeto ofereça as variáveis mais favoráveis, como corrente a favor do destino, vento contra o barco e assim obter o maior desempenho da embarcação.

O protótipo implementado tem como função buscar rotas cadastradas, no aplicativo, e encontrar percursos entre o ponto de partida e destino, priorizando como melhor trajeto, caso a busca pelo melhor caminho seja nula, a via de menor distância.

9.1 METODOLOGIA

Foi realizado um levantamento bibliográfico sobre os conceitos a serem aplicados para o desenvolvimento, dentre eles, teoria dos grafos, funcionamento do algoritmo de Dijkstra, funcionalidade do GPS, prioridades para uma navegação mais rápida, cartas náuticas e percursos encontrados na região de Santa Catarina e por fim a plataforma de desenvolvimento *Android* para a elaboração de um aplicativo que utilizasse os conceitos aprendidos na fundamentação teórica.

A segunda etapa foi a definição das ferramentas para desenvolvimento do protótipo. Foi optado pelo uso, do *Android*, utilizando *a integrated development environment* (IDE) *Android Studio,* pois é uma ferramenta robusta e focada no desenvolvimento para a plataforma *Android*.

A terceira etapa foi a definição do que seria utilizado para desenvolvimento do protótipo. Nesta fase foram identificadas algumas das prioridades relacionadas à navegação de barcos pequenos, com foco nos modelos a vela. Foi desenvolvido um quadro de regras para ser aplicado nas tomadas de decisão que o algoritmo de Dijkstra deverá utilizar.

Após levantamento bibliográfico e as definições acerca do protótipo e do modelo de simulação foram definidos alguns percursos da região de Santa Catarina para utilizá-los em testes para funcionamento do aplicativo.

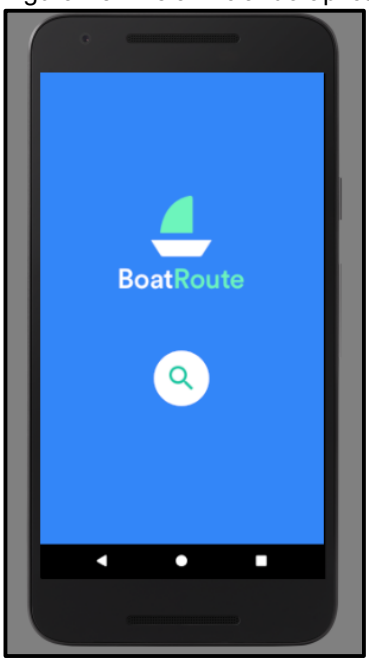
9.1.1 Desenvolvimento

O BoutRoute foi desenvolvido utilizando como ambiente o Java JDK8 Update 55, com a IDE *Android* Studio 3.0. O modelo do aparelho móvel utilizado como base de testes foi o Moto G3 com versão do *Android* 6.0. A máquina onde foi desenvolvido o protótipo possui, 8Gb RAM, 500 Gb HD, 2,4 GHz Intel Core i5 com sistema operacional Mac OS Sierra 10.12.6

O objetivo do trabalho é utilizar o algoritmo de Dijkstra com uma nova variação proposta, onde são aplicados conceitos para encontrar o melhor percurso de navegação. Como base para o teste do algoritmo foram utilizadas algumas das rotas náuticas de Santa Catarina. O cálculo estimado para o tempo de viagem foi baseado numa simulação com o valor da corrente simulada e a distância real obtida através da *API* do *Google Maps*.

Para a utilização do BoatRoute deve-se, a partir da tela inicial, clicar no botão de busca para então selecionar uma das rotas cadastradas (figura 23).

Figura 23 - Tela inicial do aplicativo BoatRoute.



Fonte: Do autor.

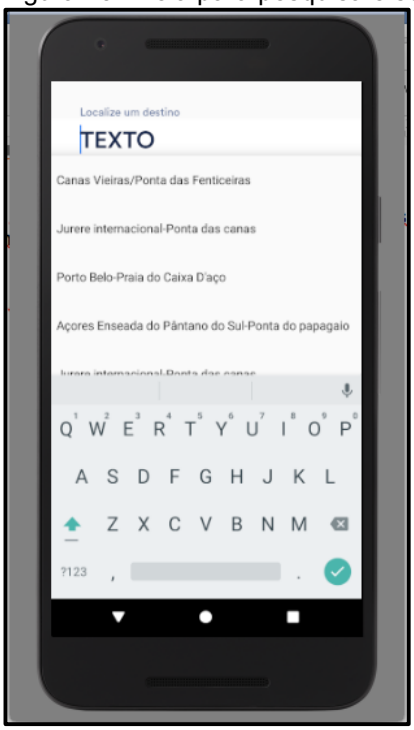
É possível a troca de rotas e velocidade pelo usuário, clicando em cima das informações exibidas referentes a esses itens (figura 24).

Figura 24 - Tela para troca de velocidade e rota desejada.

Fonte: Do autor.

Para encontrar uma rota basta digitar o nome na barra de pesquisa e confirmar a seleção da mesma clicando em cima da desejada (figura 25).

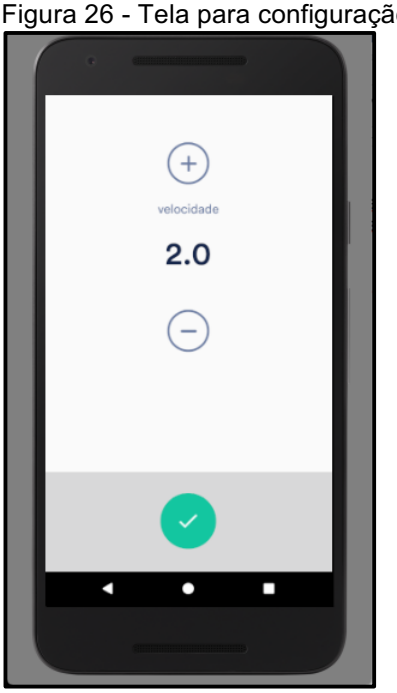
Figura 25 - Tela para pesquisa e seleção da rota desejada.



Fonte: Do autor.

O protótipo leva como parâmetro pré-configurado a velocidade de 10 km/h, entretanto, como ilustra a figura 26, o utilizador poderá acessar uma tela de configuração para alterar a velocidade conforme o modelo de seu barco.

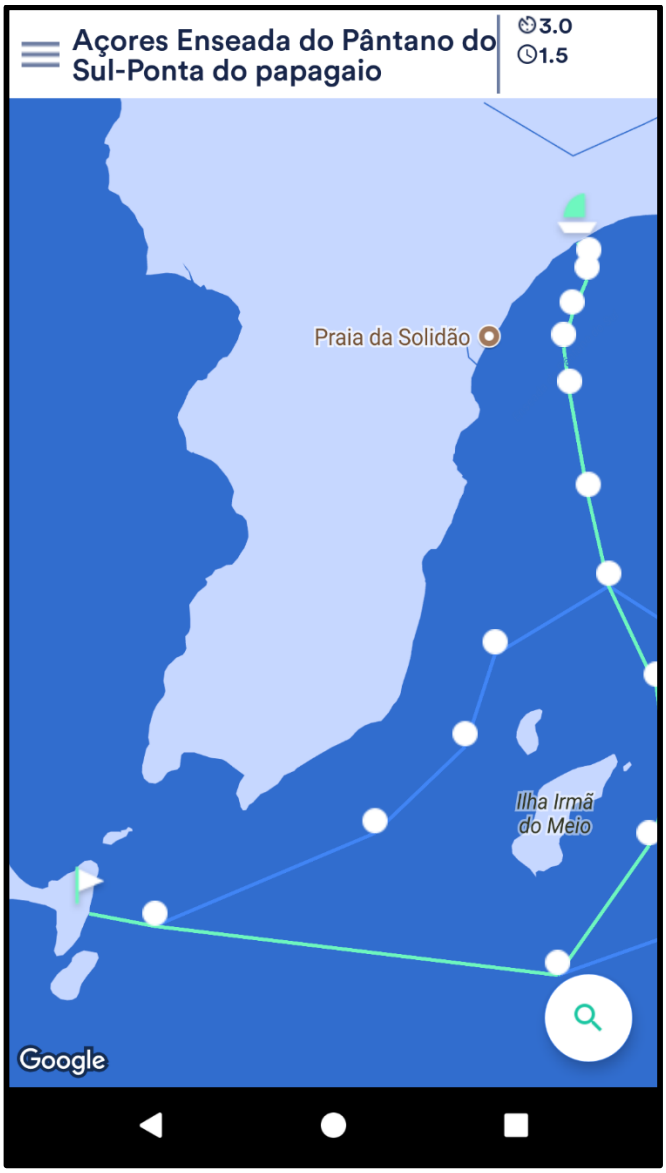
Figura 26 - Tela para configuração do range de velocidade do barco a ser utilizado.



Fonte: Do autor.

Após configurada a velocidade e selecionada uma rota de navegação poderá ser iniciado a busca pelo melhor percurso. Essa informação será apresentada em um mapa sendo destacado de verde o melhor caminho, o mais rápido. Na cor azul é informado os demais caminhos para o destino da rota desejada, conforme representado na imagem 27. Nesta tela também são encontradas as informações de tempo estimado para a navegação do melhor percurso e a velocidade média do barco que foi configurado pelo usuário.

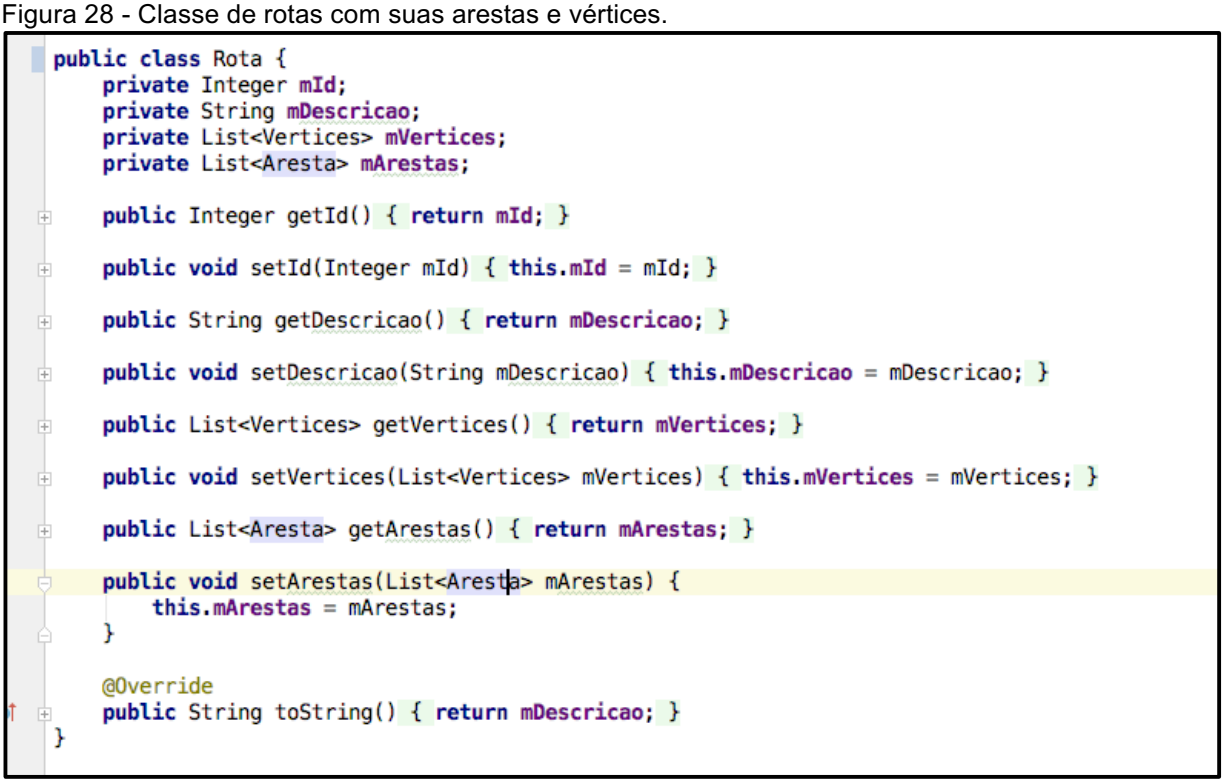
Figura 27 - Tela de representação dos percursos da rota selecionada, no canto superior mostra a velocidade media a ser atingida e tempo estimado para completar o percurso.



Fonte: Do autor.

Para a implementação da variação do algoritmo, foi necessária a criação de alguns objetos a fim de representar em forma de código as arestas e vértices de um grafo, como mostrado na figura número 28. A figura mostra a implementação de uma classe de rota onde são consumidos os objetos de arestas e vértices.

Figura 28 - Classe de rotas com suas arestas e vértices.

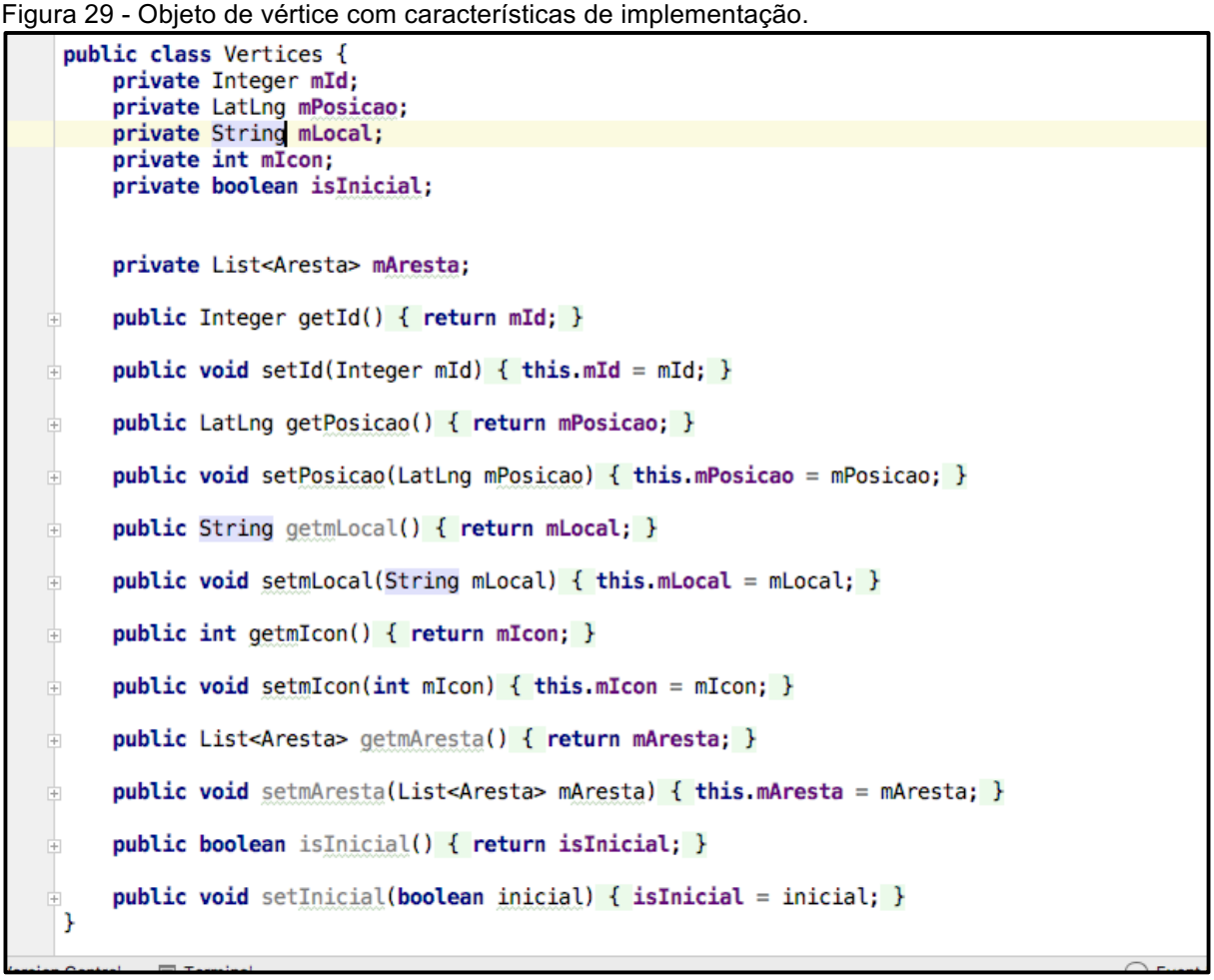


```
public class Rota {
    private Integer mId;
    private String mDescricao;
    private List<Vertices> mVertices;
    private List<Aresta> mArestas;

    public Integer getId() { return mId; }

    public void setId(Integer mId) { this.mId = mId; }

    public String getDescricao() { return mDescricao; }

    public void setDescricao(String mDescricao) { this.mDescricao = mDescricao; }

    public List<Vertices> getVertices() { return mVertices; }

    public void setVertices(List<Vertices> mVertices) { this.mVertices = mVertices; }

    public List<Aresta> getArestas() { return mArestas; }

    public void setArestas(List<Aresta> mArestas) {
        this.mArestas = mArestas;
    }

    @Override
    public String toString() { return mDescricao; }
}
```

Fonte: Do autor.

Para o objeto de vértice foram representadas em forma de variáveis, características utilizadas na tomada de decisão da variação proposta no algoritmo, como posição, localização no mapa, uma breve descrição da localidade e dentre outras conforme mostrado na figura 29.

Figura 29 - Objeto de vértice com características de implementação.

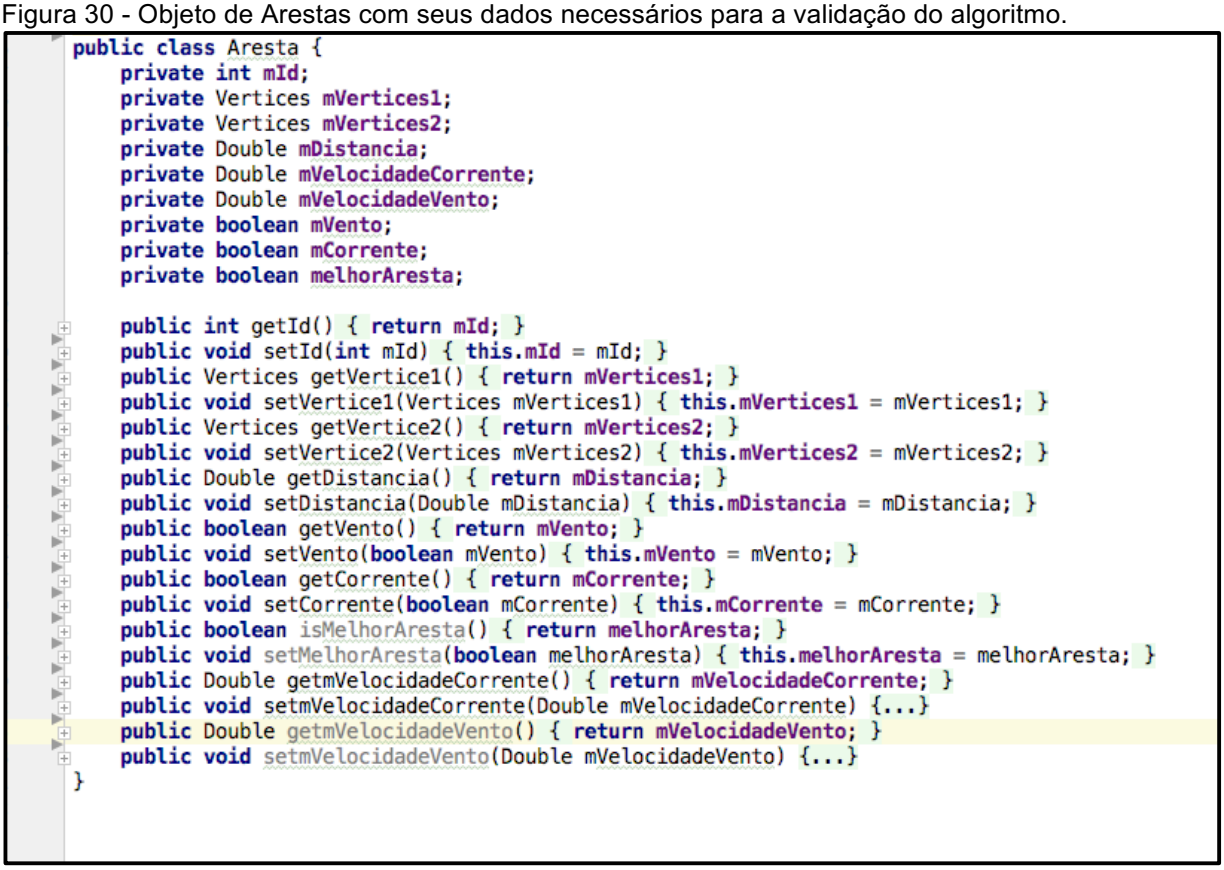


```
public class Vertices {
    private Integer mId;
    private LatLng mPosicao;
    private String mLocal;
    private int mIcon;
    private boolean isInicial;


    private List<Aresta> mAresta;

    public Integer getId() { return mId; }

    public void setId(Integer mId) { this.mId = mId; }

    public LatLng getPosicao() { return mPosicao; }

    public void setPosicao(LatLng mPosicao) { this.mPosicao = mPosicao; }

    public String getmLocal() { return mLocal; }

    public void setmLocal(String mLocal) { this.mLocal = mLocal; }

    public int getmIcon() { return mIcon; }

    public void setmIcon(int mIcon) { this.mIcon = mIcon; }

    public List<Aresta> getmAresta() { return mAresta; }

    public void setmAresta(List<Aresta> mAresta) { this.mAresta = mAresta; }

    public boolean isInicial() { return isInicial; }

    public void setInicial(boolean inicial) { isInicial = inicial; }
}
```

Fonte: Do autor.

Abstraindo as informações que devem existir nas arestas para as validações do algoritmo, foi implementado uma classe pai como mostra a figura 30. Esta classe tem como informações as distâncias das arestas, se possui vento, assim também como os vértices adjacentes a mesma.

Figura 30 - Objeto de Arestas com seus dados necessários para a validação do algoritmo.



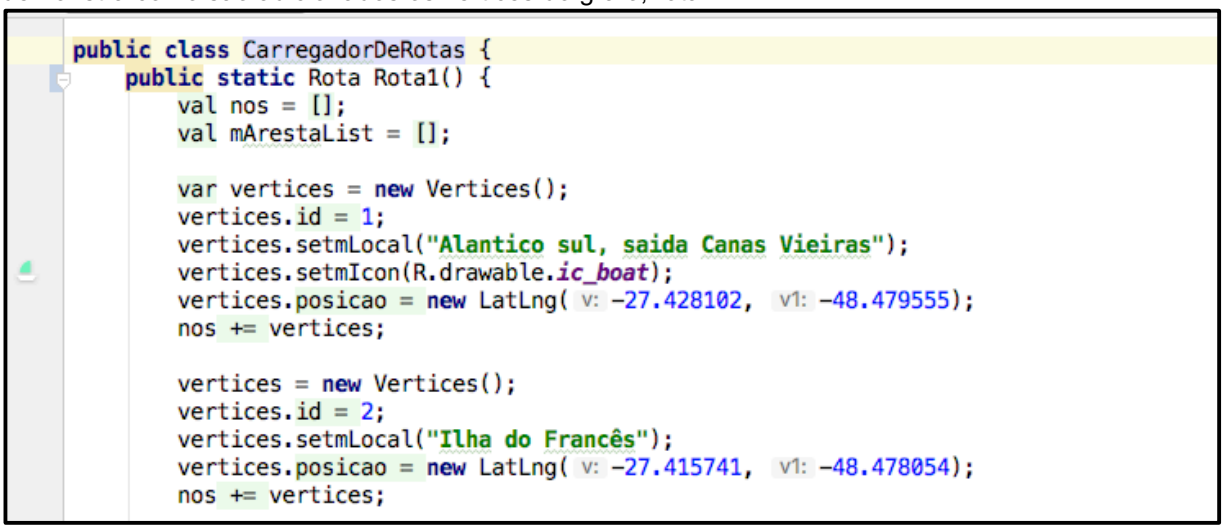
```
public class Aresta {
    private int mId;
    private Vertices mVertices1;
    private Vertices mVertices2;
    private Double mDistancia;
    private Double mVelocidadeCorrente;
    private Double mVelocidadeVento;
    private boolean mVento;
    private boolean mCorrente;
    private boolean melhorAresta;

    public int getId() { return mId; }
    public void setId(int mId) { this.mId = mId; }
    public Vertices getVertice1() { return mVertices1; }
    public void setVertice1(Vertices mVertices1) { this.mVertices1 = mVertices1; }
    public Vertices getVertice2() { return mVertices2; }
    public void setVertice2(Vertices mVertices2) { this.mVertices2 = mVertices2; }
    public Double getDistancia() { return mDistancia; }
    public void setDistancia(Double mDistancia) { this.mDistancia = mDistancia; }
    public boolean getVento() { return mVento; }
    public void setVento(boolean mVento) { this.mVento = mVento; }
    public boolean getCorrente() { return mCorrente; }
    public void setCorrente(boolean mCorrente) { this.mCorrente = mCorrente; }
    public boolean isMelhorAresta() { return melhorAresta; }
    public void setMelhorAresta(boolean melhorAresta) { this.melhorAresta = melhorAresta; }
    public Double getmVelocidadeCorrente() { return mVelocidadeCorrente; }
    public void setmVelocidadeCorrente(Double mVelocidadeCorrente) {...}
    public Double getmVelocidadeVento() { return mVelocidadeVento; }
    public void setmVelocidadeVento(Double mVelocidadeVento) {...}
}
```

Fonte: Do autor.

Com as classes pais de modelos já definidas, pode-se criar objetos para que sejam abstraídas algumas rotas fixas de navegações encontradas na região de Santa Catarina, como exemplifica a figura 31.

Para cada uma das rotas é criada uma lista de arestas e vértices e nesses são cadastrados seus dados necessários para que sejam criados os trajetos.

Figura 31-Criação de uma rota, definição da lista de arestas e vértices. Nesta imagem também demonstra como são adicionados os vértices do grafo, rota.

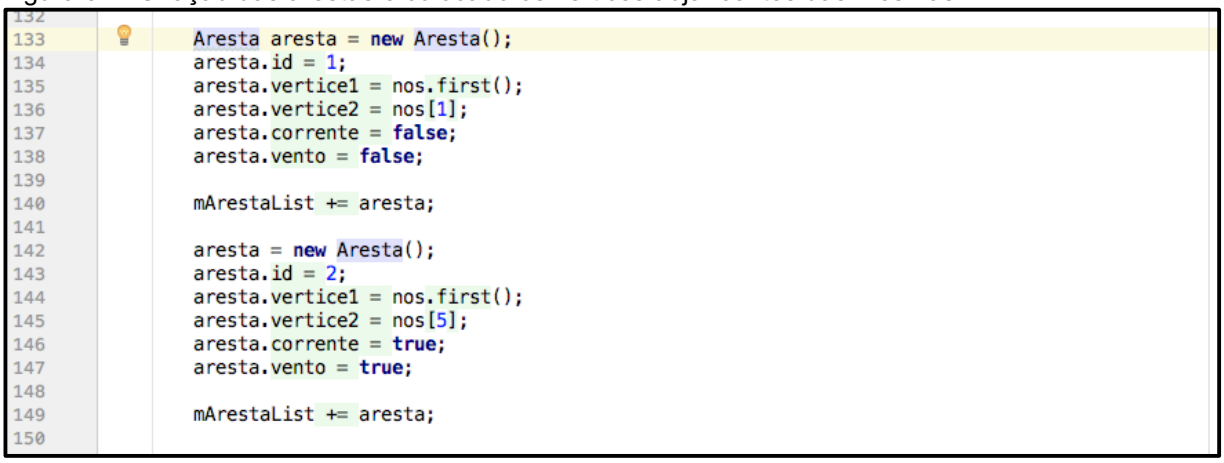


```
public class CarregadorDeRotas {
    public static Rota Rota1() {
        val nos = [];
        val mArestaList = [];

        var vertices = new Vertices();
        vertices.id = 1;
        vertices.setmLocal("Alantico sul, saida Canas Vieiras");
        vertices.setmIcon(R.drawable.ic_boat);
        vertices.posicao = new LatLng( v: -27.428102,  v1: -48.479555);
        nos += vertices;

        vertices = new Vertices();
        vertices.id = 2;
        vertices.setmLocal("Ilha do Francês");
        vertices.posicao = new LatLng( v: -27.415741,  v1: -48.478054);
        nos += vertices;
```

Fonte: Do autor.

Adiciona-se uma aresta ao grafo e nesse momento também são vinculados os vértices adjacentes de cada aresta.

Figura 32 - Criação das arestas e colocado os vertices adjancentes das mesmas.

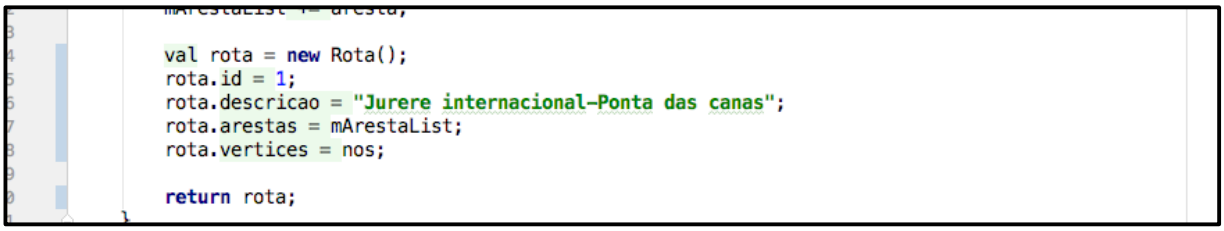


```
133     Aresta aresta = new Aresta();
134     aresta.id = 1;
135     aresta.vertice1 = nos.first();
136     aresta.vertice2 = nos[1];
137     aresta.corrente = false;
138     aresta.vento = false;
139
140     mArestaList += aresta;
141
142     aresta = new Aresta();
143     aresta.id = 2;
144     aresta.vertice1 = nos.first();
145     aresta.vertice2 = nos[5];
146     aresta.corrente = true;
147     aresta.vento = true;
148
149     mArestaList += aresta;
150
```

Fonte: Do autor.

Com a finalização dos cadastros das arestas e os vértices, as mesmas são adicionadas em um grafo, criando-se então uma rota.

Figura 33 - Finalização da criação de um percurso abstraindo-o para um grafo.



```
        mArestaList != aresta;

        val rota = new Rota();
        rota.id = 1;
        rota.descricao = "Jurere internacional-Ponta das canas";
        rota.arestas = mArestaList;
        rota.vertices = nos;

        return rota;
    }
```

Fonte: Do autor.

Ao ser selecionada uma rota no protótipo para a exibição do percurso, e o tempo estimado para a melhor rota, são seguidos uma série de validações, começando na hora de leitura do mapa como mostrado na figura 34.

Figura 34 - Momento de iniciação do mapa, onde são recebidos os parâmetros para que seja montado o grafo e cálculos necessários.

```
@Override
public void onMapReady(final GoogleMap googleMap) {


    map = googleMap;
    markerPoints = [];
    val santaCatarina = new LatLng( v: -27.588462,  v1: -48.336853);
    googleMap.moveCamera(CameraUpdateFactory.newLatLng(santaCatarina));
    googleMap.animateCamera( CameraUpdateFactory.zoomTo(  v: 8.0f ) );
    googleMap.mapStyle = MapStyleOptions.loadRawResourceStyle(applicationContext, R.raw.styles_json);
    loadDefaultNos();


    val extras = intent.extras;
    if (extras != null) {
        rotaId = Integer.parseInt(extras.getString( key: "RotaSelecionadaId"));
        velocidade = Double.valueOf(extras.getString( key: "velocidade"));
    }

    loadComponents();
    loadComponetsEvents();

    if(rotaId != 0){
        nos = mRotas[rotaId].vertices;
        mArestaList = mRotas[rotaId].arestas;


        velocidadeMedia.text = velocidade.toString();
        rotaSelecionada.text = mRotas[rotaId].descricao;

        carregarGrafo();
    }
}
```

Fonte: Do autor.

A função *carregarGrafo* é a primeira chamada no momento em que a leitura do mapa converte os parâmetros recebidos, essa função inicialmente lê todos os vértices da rota selecionada a fim de ilustrar no mapa, conforme representa a figura 35.

Figura 35 - Código onde são lidas as arestas do grafo para serem adicionadas ao mapa.

```
private void carregarGrafo() {
    mPioresresArestas = [];
    mMelhoresArestas =  [];
    map.clear();

    for (val no : nos) {

        val marker = new MarkerOptions().position(no.posicao).title(String.valueOf(no.id-1));

        if (no.mIcon!= 0) {
            marker.icon(BitmapDescriptorFactory.fromResource(no.mIcon));
        } else {
            marker.icon(BitmapDescriptorFactory.fromResource(R.drawable.no));
        }
        map.moveCamera(CameraUpdateFactory.newLatLng(no.posicao));
        map.animateCamera( CameraUpdateFactory.zoomTo( v: 11.0f ) );
        map.addMarker(marker);

    }
    buildRoute(nos.first());


}
```

Fonte: Do autor.

Após inseridas as arestas no mapa é dado início ao desenho da rota. São checadas se existem arestas na rota selecionada e caso essa informação exista, inicia-se uma função que separa as melhores arestas, as que vão indicar para o caminho mais ágil, das piores, que levam a uma trajetória mais demorada, concluindo então a determinação do melhor caminho. Esse bloco é executado até que não encontre mais arestas para serem validadas, sendo assim o algoritmo finaliza sua busca pelas melhores arestas.

Figura 36 - Código do momento em que é iniciado o desenho do grafo e a busca pelas melhores arestas.

```
// => Begin Arestas
private void buildRoute(Vertices no) {
    val arestas = [];
    arestas += getArestas(no);
    var melhorAresta = new Aresta();
    if (!arestas.empty) {
        melhorAresta = getMelhorAresta(arestas);
        mMelhoresArestas += melhorAresta;
        val proximoNo = melhorAresta.vertice2;
        if (proximoNo != null) {
            buildRoute(proximoNo);
        } else {
            finishBuildRoute();
        }
    } else {
        finishBuildRoute();
    }
}
```

Fonte: Do autor.

Para encontrar as melhores arestas é executado um bloco de repetição que a cada interação utiliza duas arestas para a comparação da melhor dentre elas.

Figura 37 - Demonstração da função getMelhorArestas.

```
private Aresta getMelhorAresta(List<Aresta> arestas) {
    var melhorAresta = arestas.first();
    for (val aresta : arestas) {
        aresta.distancia = RSUtil.CalculationByDistance(aresta.vertice1.posicao, aresta.vertice2.posicao);
        aresta.setmVelocidadeCorrente(RSUtil.radomValue(1,10));//TO DO Setar na class
        melhorAresta = getMelhorAresta(melhorAresta, aresta);
    }

    return melhorAresta;
}
```

Fonte: Do autor.

Em busca das melhores arestas é necessário uma série de validações, neste caso de duas em duas são aplicados testes conforme demonstrado na figura 38.

Para a determinação de melhores arestas foram estudados alguns conceitos para uma navegação mais ágil e com isso foram estipulados as seguintes regras a serem aplicadas. Primeiro é considerado o vértice inicial, aquele que é definido como o ponto de partida. Com essa informação o algoritmo verifica dentre as possibilidades qual o melhor vértice adjacente do vértice inicial, levando em conta as seguintes validações:

a) Verifica nas arestas quais apresentam corrente a favor para o destino da rota;
b) Valida na atual aresta se o vento está contra o destino, pois com esta variável o barco tem maior força para navegação;
c) Caso nas arestas adjacentes não exista vento contra é pego então a com corrente a favor;
d) Caso não exista nem vento contra e nem corrente a favor é considerada a aresta de menor distância.

Figura 38 - Bloco de validações das regras para serem definidos as melhores arestas.

```
private Aresta getMelhorAresta(Aresta primeira, Aresta segunda) {
    if (validaCorrente(primeira) && validaVento(primeira) && validaCorrente(segunda) && validaVento(segunda)) {
        if (primeira.distancia < segunda.distancia) {
            return primeira;
        } else {
            return segunda;
        }
    } else {
        if (validaCorrente(primeira) && validaVento(primeira) && (!validaCorrente(segunda) || !validaVento(segunda))) {
            return primeira;
        } else if (validaCorrente(primeira) && !validaVento(primeira) && (validaCorrente(segunda) || !validaVento(segunda))) {
            if (primeira.distancia < segunda.distancia) {
                return primeira;
            } else {
                return segunda;
            }
        } else if (!validaCorrente(primeira) || !validaVento(primeira) && (validaCorrente(segunda) && validaVento(segunda))) {

            return segunda;

        } else if (!validaCorrente(primeira) || !validaVento(primeira) && (validaCorrente(segunda) || !validaVento(segunda))) {
            if (primeira.distancia < segunda.distancia) {
                return primeira;
            } else {
                return segunda;
            }
        }

    }
    return primeira;
}
```

Fonte: Do autor.

Quando não encontradas mais arestas para serem validadas é chamado o método que finaliza o algoritmo, este métdo é aplicado em sobre o grafo. Nesse momento iniciam-se as funções responsáveis por destacar as melhores arestas em verde e as piores em azul, como mostra a figura 39.

Figura 39 - Função que finaliza o algoritmo e chama as funções para pintar as arestas.

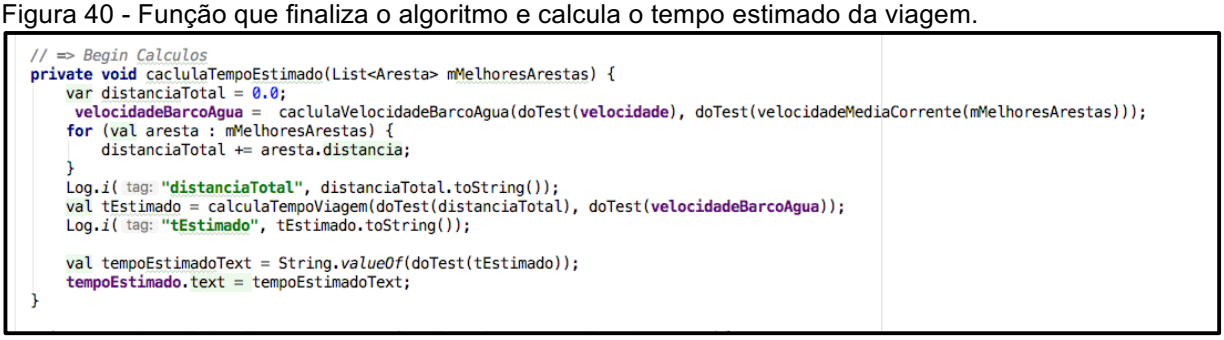
```
private void finishBuildRoute() {
    for (val aresta: mArestaList) {
        var existeMelhorAresta = false;
        for (val melhorAresta: mMelhoresArestas) {
            if (aresta.id == melhorAresta.id) {
                existeMelhorAresta = true;
            }
        }
        if (!existeMelhorAresta) {
            mPioresresArestas += aresta;
        }
    }
    for (val aresta: mPioresresArestas) {
        printPiorAresta(aresta);
    }

    for (val aresta: mMelhoresArestas) {

        printAresta(aresta);

    }
    caclulaVelocidadeMedia(mMelhoresArestas);

}
```

Fonte: Do autor.

Com o melhor percurso determinado pela aplicação do algoritmo, também é calculado uma estimativa de tempo para a navegação deste percurso. Para calcular esta estimativa de tempo busca-se a velcidade média do barco barco cadastrado em relação a corrente.

A velocidade média do barco deve ser informada para o aplicativo, no momento em que o usuário irá selecionar uma rota, para que o cálculo seja o mais assertivo possível em relação ao tempo e a velocidade média do modelo do barco usado como demostra a função apresentada na figura 40. Para a velocidade da corrente da água foram simulados números randômicos de 1 a 10. Existe a possibilidade de se obter estes dados através de uma API, porém neste aplicativo optou-se por usar números randômicos por questão de tempo para o desenvolvimento da ferramenta.

Figura 40 - Função que finaliza o algoritmo e calcula o tempo estimado da viagem.

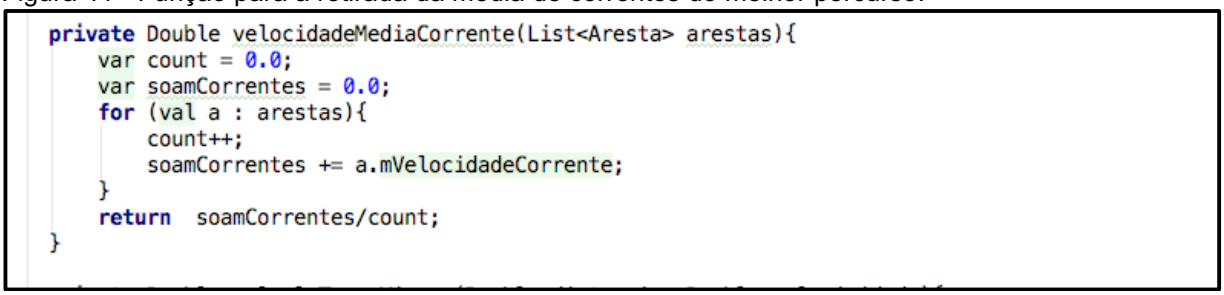


```
// => Begin Calculos
private void caclulaTempoEstimado(List<Aresta> mMelhoresArestas) {
    var distanciaTotal = 0.0;
     velocidadeBarcoAgua =  caclulaVelocidadeBarcoAgua(doTest(velocidade), doTest(velocidadeMediaCorrente(mMelhoresArestas)));
    for (val aresta : mMelhoresArestas) {
        distanciaTotal += aresta.distancia;
    }
    Log.i( tag: "distanciaTotal", distanciaTotal.toString());
    val tEstimado = calculaTempoViagem(doTest(distanciaTotal), doTest(velocidadeBarcoAgua));
    Log.i( tag: "tEstimado", tEstimado.toString());

    val tempoEstimadoText = String.valueOf(doTest(tEstimado));
    tempoEstimado.text = tempoEstimadoText;
}
```

Fonte: Do autor.

Para utilização nos cálculos é retirado de todas as arestas uma média do valor da velocidade das correntes encontradas no melhor percurso, conforme ilustrado na figura 41.

Figura 41 - Função para a retirada da média de correntes do melhor percurso.

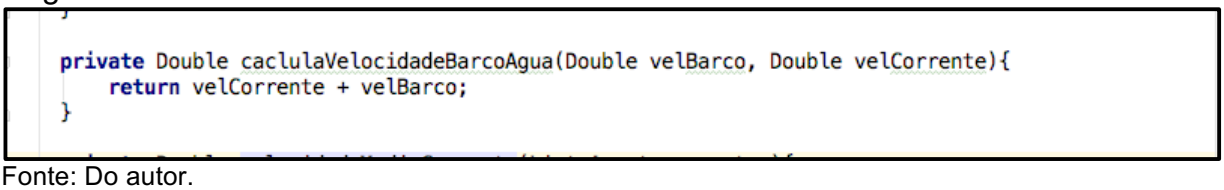


```
private Double velocidadeMediaCorrente(List<Aresta> arestas){
    var count = 0.0;
    var soamCorrentes = 0.0;
    for (val a : arestas){
        count++;
        soamCorrentes += a.mVelocidadeCorrente;
    }
    return  soamCorrentes/count;
}
```

Fonte: Do autor.

Por fim é aplicado, como mostra a figura 42, a fórmula de velocidade vetorial para estimar um tempo de navegação dentro da melhor rota apresentada pela variação do algoritmo.

Figura 42 – Aplicação do cálculo vetorial para calcular a estimativa de tempo da viagem.



```
private Double caclulaVelocidadeBarcoAgua(Double velBarco, Double velCorrente){
    return velCorrente + velBarco;
}
```

Fonte: Do autor.

10 CONCLUSÃO

Para a pesquisa foi necessário o estudo do algoritmo de caminho mínimo, neste caso foi utilizado o algoritmo de Dijkstra, obtendo assim conhecimento de técnicas que pudessem influenciar na tomada de decisão do algoritmo afim de encontrar o melhor ou o menor percurso em uma rota náutica.

Buscou-se durante o projeto encontrar formas de calcular o melhor ou menor percurso, que possibilitassem a busca destas informações e estimassem saber a velocidade média que o barco pode atingir, em movimento na água, e com essa análise apurar quanto tempo, aproximadamente, o barco levaria para percorrer este caminho.

Foi importante o entendimento da funcionalidade dos receptores GPS oferecidos pelos dispositivos móveis com sistema operacional *Android* e para esse item foi identificado que o mesmo adota de forma nativa do aparelho trabalhar em conjunto de recursos da API do Google Maps.

O conhecimento da linguagem de programação Java e seu comportamento utilizando os recursos oferecidos pela plataforma *Android* foram de grande importância para o desenvolvimento do aplicativo *BoatRout* e na aplicação dos conceitos de navegação para a tomada de decisão do algoritmo de Dijkstra.

Com o aplicativo proposto neste trabalho foi possível encontrar em rotas cadastradas, os percursos mais rápidos, caso não seja encontrado o melhor trajeto, o aplicativo informou o caminho de menor distância. O protótipo conseguiu relatar a velocidade média que o barco poderia alcançar durante a navegação e estimar o tempo da viagem.

Em busca dos melhores caminhos foram aplicados como conceitos os seguintes requisitos:

a) Verificação das arestas e quais apontam corrente a favor para o destino da rota;
b) Comparação da atual aresta se o vento está contra o destino, porque com esse requisito o barco tende a ter maior força de navegação;

c) Se na Aresta adjacentes atual não houver vento contra é pego então a que apresenta corrente a favor;
d) Na hipótese de não existir vento contra e nem corrente a favor, é considerada a melhor aresta aquela de menor distância.

Propõe-se como trabalhos futuros, formas de armazenamento das rotas em uma base de dados na nuvem. Propõe-se também uma forma de cadastro das rotas dinamicamente e compartilhamento entre os usuários para a utilização dessas rotas.

Pode-se dar continuidade a este trabalho de conclusão realizando uma pesquisa que encontre formas de utilizar a localização do GPS e recalcular a rota conforme for navegada. Outro recurso interessante à ser aplicado após o recurso de estimativa em tempo de navegação e o cadastro dinâmico e compartilhado dos usuários, seria em relação a itens como acidentes, tempestades ou outros que podem prejudicar o trajeto. Este recurso poderá ser adicionado na variação do algoritmo de Dijkstra proposto neste projeto.

REFERÊNCIAS

ALOISE, D. J; CRUZ, J. S. da. **Teoria dos grafos e aplicações**. Centro de Ciências Exatas e da Terra, Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, 2001. Disponível em: http://www.dimap.ufrn.br/~dario/arquivos/Cap1_Grafos-2001.pdf. Acesso em: 06 jun. 2008.

BESSONE; RIBEIRO; GONÇALVES, Tânia; RIBEIRO, Gladys Sabina; GONÇALVES, Monique de Siqueira. **Cultura escrita e circulação de impressos no Oitocentos**. São Paulo: Alameda, 2016.

BORTOT, José Carlos. SysNavSailBoat – **Sistema de Navegação Marítima, comInteligência Artificial, utilizando Otimização de Múltiplos Objetivos e AlgoritmosGenéticos**. 2009. 10 f. Tese (Doutorado) - Curso de Ciências da Computação, Fatec -Faculdade de Tecnologia de São Paulo, Unidades de Itaquaquecetuba e Mogi das Cruzes –São Paulo – Brasil., Itaquaquecetuba e Mogi das Cruzes, 2009. Disponível em: <http://www.centropaulasouza.sp.gov.br/pos-graduacao/workshop-de-pos-graduacao-e-pesquisa/anais/2009/trabalhos/gestao-e-desenvolvimento-de-tecnologias-da-informacao-aplicadas/trabalhos-completos/bortot-jose-carlos.pdf>. Acesso em: 18 nov. 2015.

BRASIL, Marinha do. **Cartas Náuticas Raster e Arquivos GeoTIFF**. Disponível em: <http://www.mar.mil.br/dhn/chm/box-cartas-raster/raster>. Acesso em: 05 jun. 2017.

CAMPOS, Vânia Barcellos G. **Algoritmos para Resolução de Problemas em Redes**. 2007. 30 f. Dissertação (Mestrado) - Curso de Engenharia de Transportes, Instituto Militar de Engenharia, Rio de Janeiro, 2007. Disponível em: <http://aquarius.ime.eb.br/~webde2/prof/vania/apostilas/Apostila-Redes.pdf>. Acesso em: 21 out. 2015.

COELHO, Alex. **Utilização do algoritmo de dijkstra para resolver o problema do caminho mínimo em mapas construídos com o formato scalable vector graphics**. 2004. 61 f. Tese (Doutorado) - Curso de Sistemas de Informação, Centro Universitario Luterano de Palmas, Palmas, 2004

COSTA, Carolina. **9 aplicativos úteis para marítimos.** 2015. Disponível em: <http://jornalcanal16.com.br/site/pt/pt/9-aplicativos-mais-uteis-pelos-maritimos/>. Acesso em: 19 out. 2015.

DEVELOPER, Android. **Como baixar o Android Studio e o SDK Tools**. 2015. Disponível em: . Acesso em: 26 out. 2015.
FARIA, Caroline. **GPS (Sistema de Posicionamento Global).** 2016. Disponível em: <http://www.infoescola.com/cartografia/gps-sistema-de-posicionamento-global/>. Acesso em: 11 maio 2017.

FERNANDES, Elson. **CARTA NÁUTICA - PRA QUE SERVE?** 2012. Disponível em: <http://comandante-mucuripe.blogspot.com.br/2012/11/carta-nautica-pra-que-serve.html>. Acesso em: 25 maio 2017.

GEODUC. **Mais de 70 satélites de posicionamento global já estão em órbita**. Entenda. 2015. Disponível em: <http://www.geoeduc.com/mais-de-70-satelites-de-posicionamento-global-ja-estao-em-orbita-entenda/>. Acesso em: 25 maio 2017.

GOK, Nizamettin; KHANNA, Nitin. **Building Hybrid Android Apps with Java and JavaScript**. Sebastopol: O'reilly Media, 2013. 156 p.

GUIZZO, Engelmann; ZANETTE, Simões, Silva, Scarpato. **Utilização do Algoritmo de Dijkstra para Cálculo de Rotas no Trabalho Público do Município de Criciúma/SC.2014**. Disponível em:<http://periodicos.unesc.net/sulcomp/article/viewFile/1815/1717>. Acesso em: 08 junho 2017.

HOWELL, Elizabeth. **Navstar: GPS Satellite Network.** 2013. Disponível em: <https://www.space.com/19794-navstar.html>. Acesso em: 05 maio 2017.

IBGE; GEOCIÊNCIAS, **Diretoria de. Atlas geográficos das zonas costeiras e oceânicas do Brasil.** Rio de Janeiro: Ibge, 2011.

JACKSON, Wallace. **Android Apps for Absolute Beginners**. Nova York: Apress, 2011. 344 p.

LECHETA, Ricardo R**.Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. 3ª.ed., São Paulo: Novatec Editora, 2013.

LUCKOW, Décio Heinzelmann; MELO, Alexandre Altair de. **Programação Java para a web**. São Paulo: Novatec, 2010. 638 p

KRZYSZTOF, R. Apt∗. **Edsger Wybe Dijkstra (1930–2002):A Portrait of a Genius.** 2012. Disponível em: <https://homepages.cwi.nl/~apt/ps/dijkstra.pdf>. Acesso em: 25 maio 2017.

MOREIRA, Nicoly. **Pesquisa revela crescimento de plataformas móveis**. Leia Já, abr.
2013. Disponivel em:<http://www.leiaja.com/tecnologia/2013/pesquisa-revelacrescimento-deplataformas-mov eis/> Acesso em: 29/11/2015.

MORIMOTO, Carlos E**. Smartphones, Guia Prático: GPS.** 2009. Disponível em: <http://www.hardware.com.br/livros/smartphones/capitulo-gps.html>. Acesso em: 25 maio 2017.

NETTO, Paulo Oswaldo Boa Ventura. **Grafos: Teoria, modelos, algoritmos**. São Paulo: Bluncher, 2003.

NETTO; JUTKIEWICZ, Paulo Oswaldo Boa Ventura; JURKIEWICZ, Samuel**. Grafos: Introdução e Prática**. São Paulo: Bluncher, 2011.

NICOLAU. **Os Fundamentos da Física: Cinemática vetorial (III)**. 2013. Disponível em: <http://osfundamentosdafisica.blogspot.com.br/2013/05/15-aula-cinematica-vetorial-borges-e.html>. Acesso em: 18 jun. 2017.

NICOLETTI, Maria O Carmo; HRUSCHKA JUNIOR, Estevam Rafael. **Fundamentos da teoria dos grafos para computação.** Sao Carlos: Edufscar, 2013.

PEREIRA, J.m.s.simões. **Grafos e Redes. Teoria e Algoritmos Básicos.** São Paulo: Editora Interciência, 2014.

RIBEIRO, Danilo Chagas. **Uma costa de respeito.** 2007. Disponível em: <http://www.popa.com.br/diarios/POA-FLO/>. Acesso em: 10 nov. 2017.

RK, Arjun; REDDY, Pooja; SHAMA. **RESEARCH ON THE OPTIMIZATION OF DIJKSTRA'S ALGORITHM AND ITS APPLICATIONS.** 2015. 6 f. Tese (Doutorado) - Curso de School Of Computer Science, Vellore Institute Of Technology, Tamil Nadu, 2015.

SILVA, André Luís; OLIVEIRA, Lucas Machado de; ANDRADE, Rafael Quintão de. **Metodologia baseada no Algoritmo de Dijkstra para roteirização de pontos turísticos em Ouro Preto.** 2010. 1 v. Tese (Doutorado) - Curso de Engenharia de Produção, Universidade Federal de Viçosa, Viçosa, 2010.

SILVA, Inara Soldera Romano da. **UTILIZAÇÃO DE UM ALGORITMO DE CAMINHO MÍNIMO NO PROCESSO DE RECOLHIMENTO DO PALHIÇO DA CANA-DE- AÇÚCAR**. 2009. 83 f. Dissertação (Mestrado) - Curso de Faculdade de Ciências Agronômicas, Universidade Estadual Paulista "Júlio de Mesquita

Filho” Faculdade de Ciências Agronômicas Campus de Botucatu, Botucatu, 2009. Cap. 4.

SILVA, Elton Douglas. **A OTIMIZAÇÃO DE PROBLEMAS DE CAMINHO MAIS CURTO ENTRE DOIS PONTOS EM UMA ROTA, UTILIZANDO ALGORITMOS BASEADOS EM GRAFOS**. 2012. 60 f. Tese (Doutorado) - Curso de Analise de Sistemas, Faculdade Norte Capixaba de SÂo Mateus, SÃo Mateus, 2012. Cap. 2. Disponível em: <http://saomateus.multivix.edu.br/wp-content/uploads/2013/05/A-otimizacao-de-problemasde-caminho-mais-curto-entre-dois-pontos-em-uma-rota-utilizando-algoritmos-baseados-emgrafos.pdf>.Acesso em: 28 out. 2015.

WOLFE, **Joe. The physics of sailing**. 2002. Disponível em: <http://newt.phys.unsw.edu.au/~jw/sailing.html>. Acesso em: 10 maio 2017.

ZHAN, F. Benjamin. **Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures.1997**. Disponível em:<https://pdfs.semanticscholar.org/7363/2b3e0f666e59a124447c3e906ca180ccb045.pdf>. Acesso em: 08 junho 2017.

APÊNDICE(S)

APÊNDICE A – Artigo

Boatroute: uso do algoritmo dijkstra a fim de determinar o melhor caminho para percursos náuticos na região de santa catarina

Helder R. Silva[1], Christine Vieira[2]

[1] Academico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma– SC

[2]Professora do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias – Universidade do Extremo Sul Catarinense (UNESC) – Criciúma– SC

hedie.strokes@gmail.com, cvi@unesc.net

Abstract. *The use of technology diffuses daily for the automation of various activities, in the nautical area it has much to add. The project proposes the combination of the Dijkstra algorithm and applied concepts to achieve faster navigation in order to offer an application that can aid in the process of searching for more agile routes. For the definition of concepts and application of the application was carried out bibliographical survey and a search for development in order to offer a prototype that estimated the best routes of a route. The project brought positive results generating the BoatRout prototype being possible to find better routes estimating the average time for the routes.*

Resumo. *O uso da tecnologia se difunde diariamente para a automatização de diversas atividades, na área náutica ela tem muito a agregar. O projeto propõe a combinação do algoritmo de Dijkstra e conceitos aplicados para que se atinja uma navegação mais rápida a fim de oferecer um aplicativo que possa auxiliar no processo de busca por percursos mais ágeis. Para a definição de conceitos e aplicação do aplicativo foi realizado levantamento bibliográfico e uma busca por desenvolvimento a fim de oferecer um protótipo que estimasse os melhores percursos de uma rota. O projeto trouxe resultados positivos gerando o protótipo BoatRout sendo possível encontrar melhores percursos estimando a média de tempo para as rotas.*

1. Introdução

O uso da tecnologia se expande diariamente em prol da simplificação de problemas de determinadas áreas, agilizando ou automatizando serviços. Como em quaisquer outras atividades dentro da cultura de navegação o uso de inovações tecnológicas vem para agregar. Muitos softwares estão disponíveis para uso dentro da área naval, porém a grande maioria necessita do auxílio de equipamentos para melhor funcionamento.

Entre a aparelhagem necessária, podemos encontrar micro controladores, sinais dos satélites do *Global Positioning System* (GPS), dentre outros. Entre as que ajudam navegadores de pequenas embarcações, destacam-se computadores de bordo que

tem como função informar o posicionamento do barco em mapas digitais, direção, e outras informações úteis para auxiliar em uma navegação segura e inteligente. O uso de muitos equipamentos em pequenas embarcações, como veleiros, por exemplo, torna-se difícil. Mesmo que estes barcos sejam de um tamanho maior, o investimento de equipagem torna-se alto (BORTOT, 2009).

Visando buscar soluções relacionadas ao espaço físico, o desenvolvimento de aplicativos *mobile* se difunde gradativamente. O uso de *smartphones* está mais presente no cotidiano, e a quantidade de aplicativos para os dispositivos, com diversas funcionalidades e objetivos, cresce no mesmo ritmo. Aplicativos voltados para a navegação marítima em aparelhos móveis estão sendo lançados e abrangem desde o planejamento de rotas à orientações de usuários comuns. Mesmo o utilizador podendo determinar qual aplicativo irá usar em seu celular, com foco em geonavegação, em muitos casos acaba não utilizando um software de auxílio, pois em grande parte os aplicativos disponíveis não são gratuitos para download ao público alvo (COSTA, 2015).

Com o uso da tecnologia GPS integrada no *smartphone*, foram desenvolvendo-se aplicativos úteis para descoberta de caminhos e rotas. Alguns programas que auxiliam os navegadores utilizando inovação são: Garmin Mobile XT, Maplink Destinator, Route 66, BeeLineGPS. Eles tendem a auxiliar embarcações náuticas na estimativa de menores rotas, mostrando o posicionamento do barco em um mapa, entre outras funções que possam beneficiar os navegadores (MORIMOTO, 2009).

Sabendo da existência de aplicações já desenvolvidas, pode-se encontrar as mesmas disponíveis a partir de tarifas pagas ou disponibilizadas em versões *trials*, gratuitas apenas para testes de curto prazo. Dentre os valores dos investimentos, os preços iniciais para aplicativos náuticos na *PlayStore* alternam-se de R$ 11,00 (onze reais) a R$ 150 (cento e cinquenta reais). No entanto, muitos dos usuários de embarcações de pequeno porte são considerados hobbista, a maioria opta por não comprar e consumir este tipo de softwares devido ao investimento necessário.

Perante a situação apresentada, referindo-se ao custo para a acessibilidade dos softwares de apoio em navegações marítimas, o objetivo desta pesquisa é oferecer uma nova solução gratuita, criando um aplicativo a partir de tecnologias livres e usando o algoritmo de Dijkstra para o cálculo da melhor rota a ser seguida pela embarcação, sendo este um dos algoritmos mais utilizados para determinação de percursos mínimos. O aplicativo também irá utilizar o GPS integrado à um dispositivo que contenha o sistema operacional *Android*. O aplicativo proposto tende a sanar o planejamento de rotas, e buscar o percurso mais rápido, caso não encontre a rota mais ágil oferecer a rota de melhor ou menor distância.

O protótipo proposto prioriza a busca por rotas mais rápidas ou as de menor distância, levando em consideração informações como o vento, correnteza dentre outras variáveis necessárias para a otimização da navegação. Além disso utilizando os recursos oferecidos pelo GPS para que nos forneça uma distância para um percurso. O protótipo a ser construído será disponibilizado na forma de código-aberto, gratuitamente e com o instalador numa conta do *GitHub*, com o intuito de difundir softwares móveis para apoio à navegação marítima no estado de Santa Catarina, buscando mapear pontos náuticos recomendados para tráfego marítimo.

2. Caminho Mínimo

Netto e Jurkiewicz (2011), dizem que na teoria dos grafos, o problema do caminho mínimo tende na minimização para percorrer um grafo entre dois vértices.

Este problema pode ser modelado através de um grafo, aonde os pontos de entrada e saída são representados pelos nodos, a cada um deles sendo associado um nome. As arestas correspondem ao caminho entre os pontos, sendo que o caminho deve ser percorrido apenas pela direção das flechas de cada aresta, adicionalmente aparece um peso associado a cada aresta que corresponde ao setor do caminho indicado possui um peso que corresponde. O custo do caminho é dado pela soma de cada aresta percorrida e pelo cálculo destes valores, existem alguns algoritmo que podem ser aplicados.

3 DJKISTRA

O algoritmo de Dijkstra é um dos mais conhecidos para resolução do problema de menor caminho entre dois pontos de um grafo.

3.1 EDSGAR WYBE DIJKSTRA

Edsgar Wybe Dijkstra foi um cientista da computação nascido na Holanda em 11 de maio de 1930. Filho de mãe matemática e um pai químico, o mesmo graduou-se em Matemática e Teoria da Física em 1956 na Universidade de *Leinden* e mais tarde, em 1959, recebeu seu título de PhD pela Universidade de Amsterdam com sua tese intitulada, '*Communication with an Automatic Computer*', que em português ficaria, '*A Comunicação com Computadores Automáticos*', na qual se dedicava a descrever a montagem da linguagem projetada para o primeiro computador comercial desenvolvido na Holanda, o X1. Edgar posteriormente veio a ser conhecido por sua contribuição à Teoria dos Grafos com o algoritmo de Dijkstra (KRZYSZTOF, 2002, tradução nossa).

Dijkstra em 1959, após estudos, propôs um algoritmo para resolver dois problemas de conexões em grafos: o menor caminho e árvore geradora mínima com fonte única. O problema dos caminhos mínimos é um clássico tema em otimização combinatória. Sendo um grafo G com pesos não negativos em suas arestas (grafo ponderado) e um vértice fonte s, encontrar caminhos (de pesos) mínimos de s a todos os demais vértices. Dado um grafo com arestas ponderadas, o problema da árvore geradora mínima baseia-se em encontrar uma árvore em que todos os vértices se conectem (direta ou indiretamente) uns aos outros. A estrutura deve possuir o menor peso possível, onde o peso é dado pela soma dos valores das arestas escolhidas (mínima) (SILVA, 2009).

O algoritmo de Dijkstra é um algoritmo que se fundamenta em localizar o caminho mínimo entre um vértice inicial e um vértice escolhido do grafo como ponto final de uma trajetória, não necessariamente que este caminho de custo mínimo passará por todos os vértices (nós) do grafo (NETTO; JUTKIEWICZ, 2011).

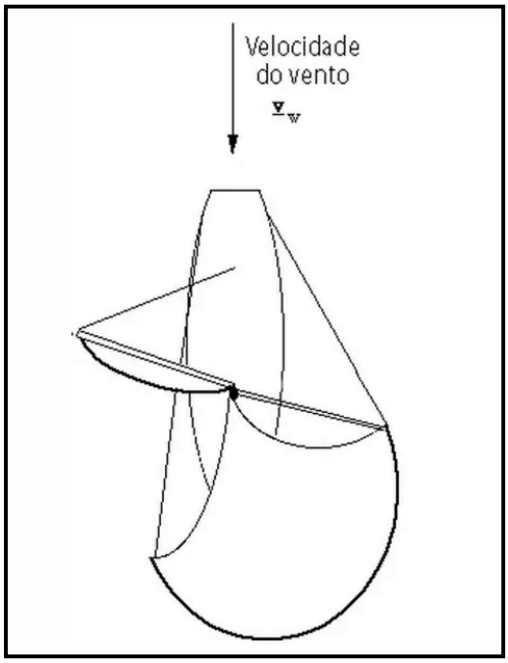
4. COMO SÃO ESTIMADOS OS RECURSOS NÁUTICOS

Para que seja estimado uma rota rápida os navegadores planejam a partir de variáveis climáticas que podem aparecer em seus percursos. Dentre os itens dos quais os navegadores necessitam ter maior conhecimento, temos correnteza e vento sendo os principais variáveis.

Referindo-se ao vento, encontramos duas teorias para navegação sendo estas, o veleiro pode navegar com o vento a favor da vela, conforme ilustrado na figura 1, onde, o vento sopra nas velas e desloca o barco. Nesta situação, o vento é mais rápido do que o barco, de forma que o ar é desacelerado pelas velas assim empurrando utilizando a força gerada para movimentar o barco. Entretanto um barco que navega a favor do vento sempre possui uma menor velocidade, mas nesta situação é mais confortável para o navegador pois o barco ganha

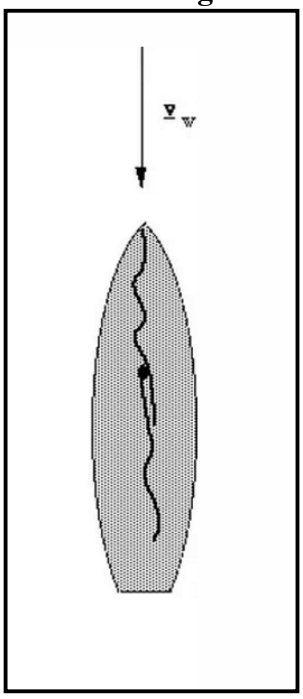
maior estabilidade, não ocorrendo grande oscilação de movimentos. Mas não é o mais interessante na navegação quando pretende-se chegar em menor tempo. Também existe a possibilidade de navegar contra o vento, os barcos a vela podem navegar contra ao vento, tendo uma aproximação de 40° em relação ao vento, para conseguir atingir esta manobra a vela tende a ficar mais fechada quase em linha reta com a ponta do barco, conforme ilustrado na figura 2 (WOLFE, 2002, tradução nossa).

Figura 43. Representação da vista de cima de um veleiro navegando a favor do vento, note-se que a vela se encontra aberta e inflada quando atingida por uma grande quantidade de vento



Fonte: Wolfe (2002).

Figura 44 - Ilustração de um veleiro com a vela aproximado da reta da ponta do barco, manobra utilizada para obter velocidade e navegar contra o vento.



Fonte: Wolfe (2002).

5 TRABALHO DESENVOLVIDO

A aplicação de conhecimentos com a pesquisa pode gerar o protótipo BoutRoute. Para sei desenvolvimento foi utilizando como ambiente o Java JDK8 Update 55, com a IDE Android Studio 3.0. O modelo do aparelho móvel utilizado como base de testes foi o Moto G3 com versão do Android 6.0. A máquina onde foi desenvolvido o protótipo possui, 8Gb RAM, 500 Gb HD, 2,4 GHz Intel Core i5 com sistema operacional Mac OS Sierra 10.12.6

O objetivo do trabalho é utilizar o algoritmo de Dijkstra com uma nova variação proposta, onde são aplicados conceitos para encontrar o melhor percurso de navegação. Como base para o teste do algoritmo foram utilizadas algumas das rotas náuticas de Santa Catarina. O cálculo estimado para o tempo de viagem foi baseado numa simulação com o valor da corrente simulada e a distância real obtida através da API do Google Maps.

5.1 VARIAÇÃO PROPOSTA PARA TOMADA DE DECISÃO DO ALGORITMO DE DIJKSTRA

Para a determinação de melhores arestas foram estudados alguns conceitos para uma navegação mais ágil e com isso foram estipulados as seguintes regras a serem aplicadas na tomada de decisão do algoritmo aplicado, Dijkistra. Primeiro é considerado o vértice inicial, aquele que é definido como o ponto de partida. Com essa informação o algoritmo verifica dentre as possibilidades qual o melhor vértice adjacente do vértice inicial, levando em conta as seguintes validações:

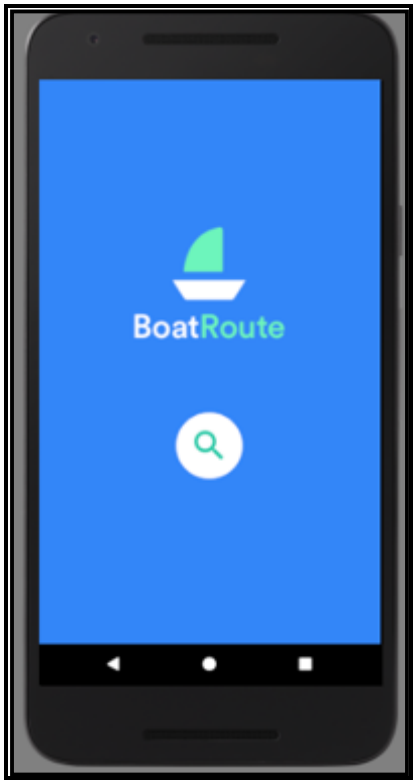
a) Verifica nas arestas quais apresentam corrente a favor para o destino da rota;
b) Valida na atual aresta se o vento está contra o destino, pois com esta variável o barco tem maior força para navegação;
c) Caso nas arestas adjacentes não exista vento contra é pego então a com corrente a favor;
d) Caso não exista nem vento contra e nem corrente a favor é considerada a aresta de menor distância.

5.2 RESULTADOS OBTIDOS

Após a definição deste conceito para variação utilizada na tomada de decisão do algoritmo de Djkistra, o mesmo foi abstraído em forma de código para ser utilizado no desenvolvimento do protótipo BoatRout, que tem como função informar ao usuário de forma ilustrativa o melhor percurso e os demais que o mesmo pode seguir.

Para a utilização do aplicativo deve-se, a partir da tela inicial, clicar no botão de busca para então selecionar uma das rotas cadastradas (figura 3).

Figura 45 - Tela inicial do aplicativo BoatRoute.



Fonte: Do autor.

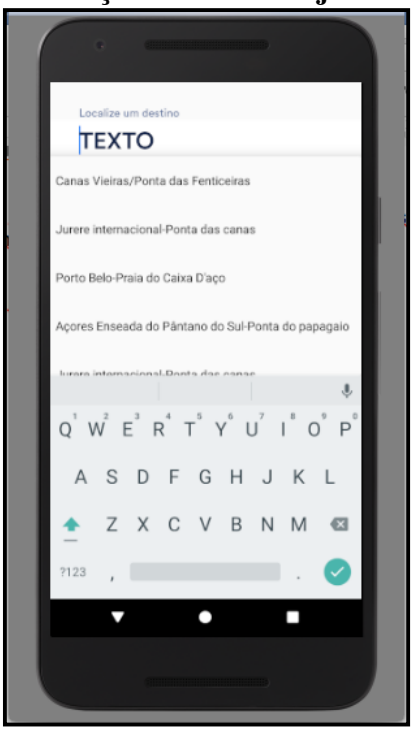
É possível a troca de rotas e velocidade pelo usuário, clicando em cima das informações exibidas referentes a esses itens (figura 4).

Figura 46 - Tela para troca de velocidade e rota desejada.

Fonte: Do autor.

Para encontrar uma rota basta digitar o nome na barra de pesquisa e confirmar a seleção da mesma clicando em cima da desejada (figura 5).

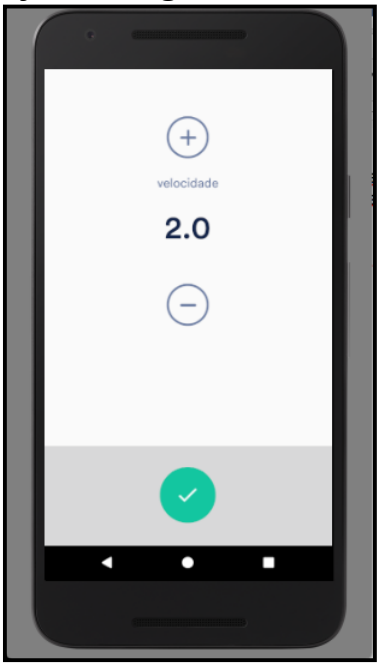
Figura 47 - Tela para pesquisa e seleção da rota desejada.



Fonte: Do autor.

O protótipo leva como parâmetro pré-configurado a velocidade de 10 km/h, entretanto, como ilustra a figura 6, o utilizador poderá acessar uma tela de configuração para alterar a velocidade conforme o modelo de seu barco.

Figura 48 - Tela para configuração do range de velocidade do barco a ser utilizado.

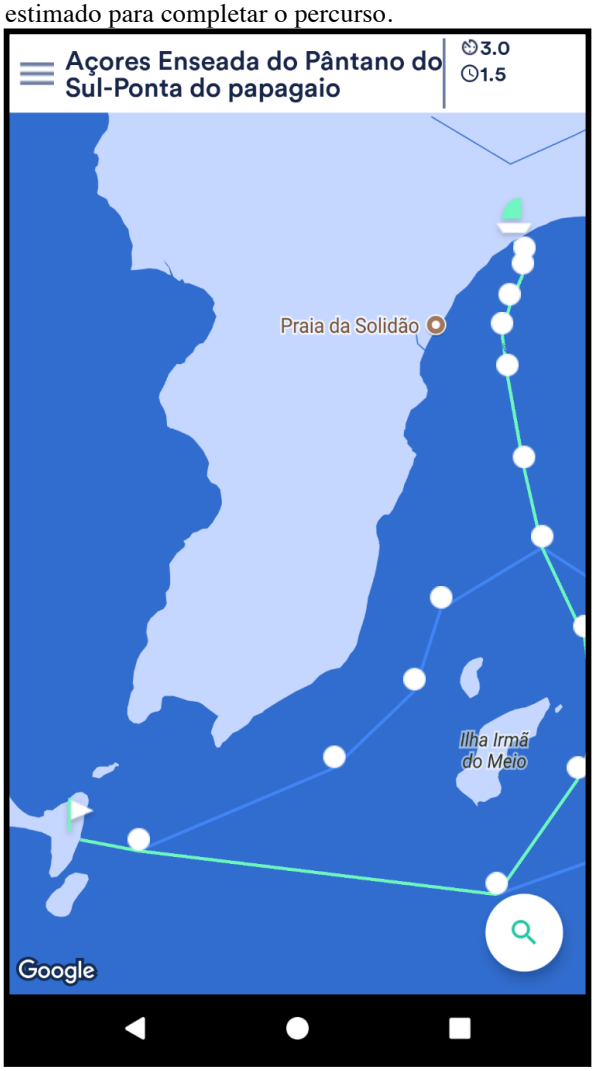


Fonte: Do autor.

Após configurada a velocidade e selecionada uma rota de navegação poderá ser iniciado a busca pelo melhor percurso. Essa informação será apresentada em um mapa sendo destacado

de verde o melhor caminho, o mais rápido. Na cor azul é informado os demais caminhos para o destino da rota desejada, conforme representado na imagem 7. Nesta tela também são encontradas as informações de tempo estimado para a navegação do melhor percurso e a velocidade média do barco que foi configurado pelo usuário.

Figura 49 - Tela de representação dos percursos da rota selecionada, no canto superior mostra a velocidade media a ser atingida e tempo estimado para completar o percurso.



Fonte: Do autor.

3. Conclusa

Para a pesquisa foi necessário o estudo do algoritmo de caminho mínimo, neste caso foi utilizado o algoritmo de Dijkstra, obtendo assim conhecimento de técnicas que pudessem influenciar na tomada de decisão do algoritmo afim de encontrar o melhor ou o menor percurso em uma rota náutica.

Buscou-se durante o projeto encontrar formas de calcular o melhor ou menor percurso, que possibilitassem a busca destas informações e estimassem saber a velocidade média que o barco pode atingir, em movimento na água, e com essa análise apurar quanto tempo, aproximadamente, o barco levaria para percorrer este caminho.

Foi importante o entendimento da funcionalidade dos receptores GPS oferecidos pelos dispositivos móveis com sistema operacional Android e para esse item foi identificado que o mesmo adota de forma nativa do aparelho trabalhar em conjunto de recursos da API do Google Maps.

O conhecimento da linguagem de programação Java e seu comportamento utilizando os recursos oferecidos pela plataforma Android foram de grande importância para o desenvolvimento do aplicativo BoatRout e na aplicação dos conceitos de navegação para a tomada de decisão do algoritmo de Dijkstra.

Com o aplicativo proposto neste trabalho foi possível encontrar em rotas cadastradas, os percursos mais rápidos, caso não seja encontrado o melhor trajeto, o aplicativo informou o caminho de menor distância. O protótipo conseguiu relatar a velocidade média que o barco poderia alcançar durante a navegação e estimar o tempo da viagem.

Em busca dos melhores caminhos foram aplicados como conceitos os seguintes requisitos:

a) Verificação das arestas e quais apontam corrente a favor para o destino da rota;

b) Comparação da atual aresta se o vento está contra o destino, porque com esse requisito o barco tende a ter maior força de navegação;

c) Se na Aresta adjacentes atual não houver vento contra é pego então a que apresenta corrente a favor;

d) Na hipótese de não existir vento contra e nem corrente a favor, é considerada a melhor aresta aquela de menor distância.

Propõe-se como trabalhos futuros, formas de armazenamento das rotas em

uma base de dados na nuvem. Propõe-se também uma forma de cadastro das rotas dinamicamente e compartilhamento entre os usuários para a utilização dessas rotas.

Pode-se dar continuidade a este trabalho de conclusão realizando uma pesquisa que encontre formas de utilizar a localização do GPS e recalcular a rota conforme for navegada. Outro recurso interessante à ser aplicado após o recurso de estimativa em tempo de navegação e o cadastro dinâmico e compartilhado dos usuários, seria em relação a itens como acidentes, tempestades ou outros que podem prejudicar o trajeto. Este recurso poderá ser adicionado na variação do algoritmo de Dijkstra proposto neste projeto.

References

BORTOT, José Carlos. SysNavSailBoat – **Sistema de Navegação Marítima, comInteligência Artificial, utilizando Otimização de Múltiplos Objetivos e AlgoritmosGenéticos**. 2009. 10 f. Tese (Doutorado) - Curso de Ciências da Computação, Fatec -Faculdade de Tecnologia de São Paulo, Unidades de Itaquaquecetuba e Mogi das Cruzes –São Paulo – Brasil., Itaquaquecetuba e Mogi das Cruzes, 2009. Disponível em: <http://www.centropaulasouza.sp.gov.br/pos-graduacao/workshop-de-pos-graduacao-e-pesquisa/anais/2009/trabalhos/gestao-e-desenvolvimento-de-tecnologias-da-informacao-aplicadas/trabalhos-completos/bortot-jose-carlos.pdf>. Acesso em: 18 nov. 2015.

COSTA, Carolina. **9 aplicativos úteis para marítimos.** 2015. Disponível em: <http://jornalcanal16.com.br/site/pt/pt/9-aplicativos-mais-uteis-pelos-maritimos/>. Acesso em: 19 out. 2015.

KRZYSZTOF, R. Apt*. **Edsger Wybe Dijkstra (1930–2002):A Portrait of a Genius.** 2012. Disponível em: <https://homepages.cwi.nl/~apt/ps/dijkstra.pdf>. Acesso em: 25 maio 2017.

MORIMOTO, Carlos E**. Smartphones, Guia Prático: GPS.** 2009. Disponível em: <http://www.hardware.com.br/livros/smartphones/capitulo-gps.html>. Acesso em: 25 maio 2017.

NETTO; JUTKIEWICZ, Paulo Oswaldo Boa Ventura; JURKIEWICZ, Samuel**. Grafos: Introdução e Prática**. São Paulo: Bluncher, 2011.

SILVA, André Luís; OLIVEIRA, Lucas Machado de; ANDRADE, Rafael Quintão de. **Metodologia baseada no Algoritmo de Dijkstra para roteirização de pontos turísticos em Ouro Preto.** 2010. 1 v. Tese (Doutorado) - Curso de Engenharia de Produção, Universidade Federal de Viçosa, Viçosa, 2010.

WOLFE, **Joe. The physics of sailing**. 2002. Disponível em: <http://newt.phys.unsw.edu.au/~jw/sailing.html>. Acesso em: 10 maio 2017.