UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

FELIPE DA SILVA PINHO

DESENVOLVIMENTO E AVALIAÇÃO DE DESEMPENHO DO SOFTWARE CAPLAN PARA PREVISÃO DE COLISÕES NO ESPAÇO AÉREO

QUIXADÁ – CEARÁ

2016

FELIPE DA SILVA PINHO

DESENVOLVIMENTO E AVALIAÇÃO DE DESEMPENHO DO SOFTWARE CAPLAN PARA PREVISÃO DE COLISÕES NO ESPAÇO AÉREO

Monografia apresentada no curso de Sistemas de Informação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação. Área de concentração: Computação.

Orientador: Arthur de Castro Callado

QUIXADÁ – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P723d Pinho, Felipe da Silva.
Desenvolvimento e avaliação de desempenho do software CAPLAN para previsão de colisões no espaço aéreo / Felipe da Silva Pinho. – 2016.
38 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2016.
Orientação: Prof. Dr. Arthur de Castro Callado.

1. Sistemas de controle de tráfego aéreo. 2. Vigilância. 3. ADS-B. 4. Software - avaliação de desempenho. I. Título.

CDD 005

FELIPE DA SILVA PINHO

DESENVOLVIMENTO E AVALIAÇÃO DE DESEMPENHO DO SOFTWARE CAPLAN PARA PREVISÃO DE COLISÕES NO ESPAÇO AÉREO

Monografia apresentada no curso de Sistemas de Informação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Sistemas de Informação. Área de concentração: Computação.

Aprovada em: Dezembro, 2016

BANCA EXAMINADORA

Arthur de Castro Callado (Orientador)
Campus Quixadá
Universidade Federal do Ceará – UFC

Rossana Maria de Castro Andrade
Departamento de Computação
Universidade Federal do Ceará - UFC

Márcio Espíndola Freire Maia
Campus Quixadá
Universidade Federal do Ceará - UFC

Este trabalho é dedicado a um pai, a uma mãe e a uma irmã esperançosa, que acreditou neste sonho que se tornou realidade.

AGRADECIMENTOS

Os agradecimentos principais são direcionados à Antônia da Silva Pinho, Carlos Luís Pinho, João Filho Costa, Etinha Lima e a todos que acrescentaram sua inestimável contribuição à minha formação pessoal e acadêmica.

Agradecimentos especiais são direcionados aos professores que fazem parte da minha história, principalmente à primeira de todas, minha irmã Lúcia Braz.

“Pode se encontrar a felicidade mesmo nas horas mais sombrias, se a pessoa se lembrar de acender a luz”

(J. K. Rowling)

RESUMO

Este trabalho apresenta uma avaliação do desempenho do software de detecção de colisão entre aeronaves que fará parte do sistema de monitoramento aéreo do projeto Radar Livre, um projeto da Universidade Federal do Ceará em seu campus de Quixadá. Este projeto baseia-se na tecnologia **ADS-B**, um método moderno de compartilhamento de informações entre aeronaves, para alimentação de seus dados. O software de previsão de colisão proposto, CAPLAN, funcionará em uma base de dados centralizada e exige um algoritmo de previsão de colisões rápido e preciso. Além disso, a implementação de tal sistema requer técnicas de otimização que o permitam processar uma grande quantidade de dados de aeronaves obedecendo a restrições de consumo de memória e de tempo de processamento que se adequem às necessidades de um sistema de previsão de colisões em tempo real. Através de pesquisas em trabalhos existentes, foi selecionado um algoritmo de previsão de colisão que analisa a iminência de colisão entre apenas duas aeronaves por vez. Tal algoritmo foi integrado ao módulo CAPLAN, restando a tarefa de aplicá-lo de forma eficiente aos dados sobre as milhares de informações de aeronaves recebidas diariamente pelo sistema. As técnicas de otimização implementadas no software são: **Multiprocessamento Simétrico**, que promove uma paralelização do processamento total do software, dividindo-o entre os processadores disponíveis no sistema; **Divisão do Espaço Aéreo em Áreas**, técnica que classifica as aeronaves em regiões e verifica a iminência de colisões apenas em áreas vizinhas, impedindo que aeronaves muito distantes sejam analisadas; e **Raio Mínimo de Verificação**, que, semelhante à técnica anterior, propõe uma distância mínima para que duas aeronaves sejam verificadas entre si na busca de colisões futuras. Cada uma dessas técnicas está detalhada da seção 4 deste trabalho. A performance do software desenvolvido foi avaliada pela técnica **Trace-based Simulation**, que é amplamente utilizada na Ciência da Computação para fazer simulações e observações durante a execução de um programa e assim medir os recursos gastos, procurar possíveis falhas e avaliar seu desempenho. Os resultados do experimento mostraram que o software é viável para compor o sistema Radar Livre desde que o mesmo trabalhe com uma quantidade máxima de 33000 aeronaves simultaneamente, o que não representa um obstáculo pois o sistema opera atualmente abrangendo a área do estado do Ceará e não recebe dados de um número de aeronaves suficiente para extrapolar esse limite.

Palavras-chave: Monitoramento Aéreo. Previsão de Colisão Entre Aeronaves. Tecnologia ADS-B.

ABSTRACT

This paper presents an evaluation of the performance of collision detection software between aircraft that will be part of the air monitoring system of the Radar Livre project, a project of the Federal University of Ceará at its Quixadá campus. This project is based on ADS-B technology, a modern method of sharing information between aircraft, to feed their data to diverse systems. The proposed collision prediction software, CAPLAN, will work in a centralized database and requires a fast and accurate collision prediction algorithm. In addition, the implementation of such a system requires optimization techniques that allow it to process a large amount of aircraft data obeying memory consumption and processing time constraints that fit the needs of a real-time collision prediction system. Through research in existing works, a collision prediction algorithm was selected that analyzes the imminence of a collision between only two aircraft at a time. This algorithm was integrated to the CAPLAN module, leaving the task of applying it efficiently to the data on the thousands of aircraft information received daily by the system. The optimization techniques implemented in the software are: **Symmetric Multiprocessing**, which promotes a parallelization of the total processing of the software, dividing it among the processors available in the system; **Division of Airspace in Areas**, a technique that classifies aircraft into regions and verifies the collision imminence only in neighboring areas, preventing very distant aircraft from being analyzed; And **Minimum Verification Radius**, which, similar to the previous technique, proposes a minimum distance for two aircraft to be verified among themselves in the search for future collisions. Each of these techniques is detailed in section 4 of this paper. The performance of the developed software was evaluated by the **Trace-based Simulation** technique, which is widely used in Computer Science to make simulations and observations during the execution of a program and thus measure the resources spent, look for possible faults and evaluate their performance. The results of the experiment showed that the software is feasible to compose the Radar Livre system since it works with a maximum quantity of 33000 aircraft simultaneously, which does not represent an obstacle because the system currently operates covering the area of the state of Ceará and does not receive enough aircraft data to exceed that limit.

Keywords: Air Monitoring. Aircraft Collision Prediction. ADS-B Technology.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FAA	Administração Federal de Aviação dos Estados Unidos
TCAS	Traffic Alert and Collision Avoidance System
TA	Traffic Advisory
RA	Resolution Advisory
ACRO	Escritório de Registros de Acidentes Aéreos
VLA	Very Light Aircraft
UFC	Universidade Federal do Ceará
ADS-B	Automatic Dependent Surveillance-Broadcast
CAPLAN	Caracara Plancus
DECEA	Departamento de Controle do Espaço Aéreo
CINDACTA	Centro Integrado de Defesa Aérea e Controle de Tráfego Aéreo
SRPV	Regional de Proteção ao Voo
ACC	Centro de Controle de Área
APP	Controle de Aproximação
TWR	Torre de Controle de Aeródromo
DTCEA	Destacamento de Controle do Espaço Aéreo
CNS/ATM	Comunicação, Navegação, Vigilância e Gerenciamento de Tráfego Aéreo
CAZ	Zona de Colisão Aeroespacial
PAZ	Zona de Proteção Aeroespacial
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
API	Interface de Programação de Aplicações
RAM	Memória de Acesso Aleatório

LISTA DE SÍMBOLOS

τ	Letra grega Tau
%	Porcentagem

SUMÁRIO

1 INTRODUÇÃO

A Administração Federal de Aviação dos Estados Unidos (FAA) está evoluindo para a próxima geração de sistemas de transporte aéreo com uma atualização completa de seus sistemas e tecnologias, visando: reduzir os atrasos, economizar combustível, diminuir a emissão de carbono e aumentar a segurança dos voos (FAA, 2015). Essa iniciativa integra novas e existentes tecnologias, incluindo a navegação por satélite e comunicações digitais avançadas.

Esta atualização surge devido ao fato de ainda se operar, mundialmente, com vários sistemas e técnicas de monitoramento aéreo ultrapassados (CHAMLOU; LOVE; MOODY, 2008). As atuais tecnologias para evitar colisões entre aeronaves estão se tornando inadequadas, especialmente frente ao aumento no tráfego aéreo mundial. O sistema mais comum para esse fim, Traffic Alert and Collision Avoidance System (TCAS), já é bastante antigo e não é capaz de acompanhar as métricas previstas para a nova geração de sistemas. É necessário, como sugerem Chamlou, Love e Moody (2008), que algumas tecnologias sejam revistas e novas soluções sejam adotadas.

O sistema TCAS, que funciona dentro das aeronaves, alerta os pilotos sobre possíveis conflitos com as aeronaves vizinhas. O TCAS I, primeira geração da tecnologia anunciada em 1981, monitora o tráfego ao redor da aeronave em um raio de 65 km e oferece informações de direção e altitude de outras aeronaves. Esta versão oferece ao piloto alertas de colisão na forma Traffic Advisory (TA), que emite um alerta sonoro sobre a proximidade de outra aeronave, porém, cabe ao piloto a resolução do conflito (WILLIAMSON; SPENCER, 1989). O TCAS II foi introduzido em 1989 e é usado na maioria dos equipamentos da aviação comercial atual. O sistema opera de forma sincronizada entre as aeronaves, gerando alertas do tipo Resolution Advisory (RA), fornecendo sugestões de mudança de rota aos pilotos com manobras verticais para evitar colisões (LEE, 2006). O sistema TCAS III foi concebido como uma extensão do sistema TCAS II, permitindo aos pilotos também manobras horizontais, além das verticais presentes nas outras versões (ROJAS; CHEN, 1989). Porém, a antena direcional utilizada no posicionamento vertical da aeronave não era precisa o suficiente, e o TCAS IV substituiu o TCAS III em meados de 1990, utilizando informações adicionais de posicionamento global para gerar uma resolução horizontal mais precisa (ZEITLIN; LOVE; CIEPLAK, 1995). O desenvolvimento do TCAS IV continuou por alguns anos, mas foi abandonado com o surgimento de novas tecnologias, tais como a Automatic Dependent Surveillance-Broadcast (ADS-B), uma tecnologia moderna e acessível desenvolvida para auxiliar o atual sistema baseado em radares no

monitoramento de aeronaves civis (ZEITLIN; LOVE; CIEPLAK, 1995).

A falta de tecnologias adequadas e a baixa precisão na detecção das colisões entre aeronaves são um grande problema para os sistemas de monitoramento aéreo. Segundo ACRO (2016), no período de 1918 até 2016 foram registradas 146.726 mortes em 23.670 acidentes aéreos, dos quais 7.625 foram causados por falha humana. Isso tende a se agravar com o aumento do tráfego de aeronaves de pequeno porte, Very Light Aircrafts (VLA), pertencentes a empresas pequenas e que operam em aeroportos secundários. Um tráfego aéreo mais denso traz a necessidade de sistemas e algoritmos mais robustos, que possam gerenciar muitas aeronaves simultaneamente (KOCHENDERFER et al., 2010).

Os algoritmos computacionais mais comuns para esse propósito, como esclarece Carbone et al. (2006), são formulados como problemas de otimização e a convergência para uma solução não é garantida em um intervalo de tempo finito e determinístico. Assim, ainda segundo Carbone et al. (2006), o algoritmo se comporta de formas diferentes para uma mesma situação de entrada de dados e não é possível prever seu tempo de execução, comportamento este exigido por um sistema de controle aéreo em tempo real.

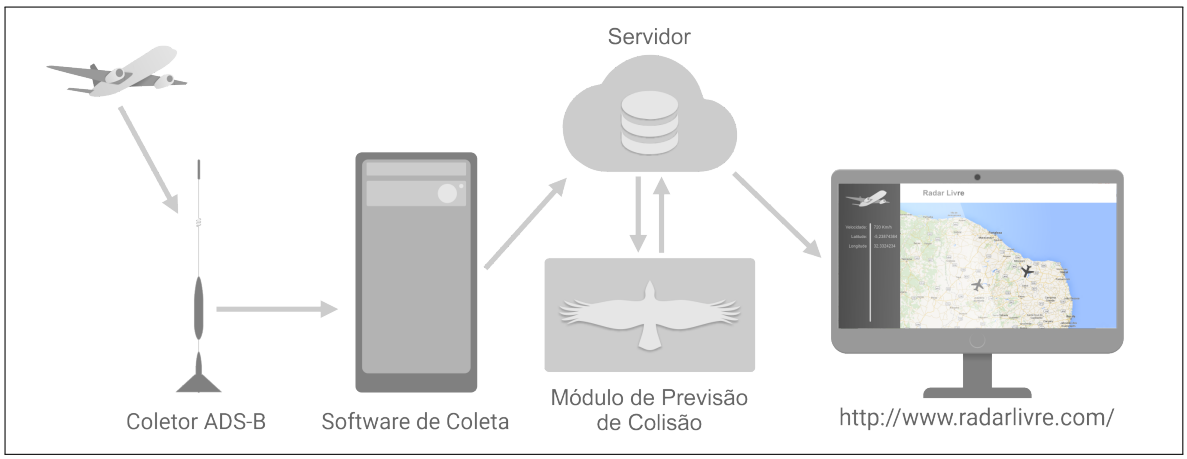
No Brasil, o problema de defasagem dos sistemas e tecnologias é semelhante. Sistemas antigos ainda operam no país, embora já existam alternativas mais baratas e confiáveis. Grande parte do nosso sistema de monitoramento aéreo ainda é baseado em radares. Assim, a eficiência do monitoramento é sujeita a falhas e a prevenção contra colisão de aeronaves é potencialmente problemática (OLIVEIRA; SALGADO, 2006).

Para auxiliar o sistema aéreo brasileiro a acompanhar os avanços tecnológicos da FAA, um grupo de pesquisa da Universidade Federal do Ceará (UFC) em Quixadá está desenvolvendo o Sistema de Monitoramento Aéreo Radar Livre, uma solução mista de hardware e software baseada na tecnologia Automatic Dependent Surveillance-Broadcast (ADS-B), a qual alimenta o sistema com dados de aeronaves em tempo real proporcionando uma base de dados centralizada e uma interface de monitoramento onde as informações podem ser visualizadas graficamente. O sistema opera fora das aeronaves, em um servidor Web, sendo mais adequado ao uso de controladores de voo, e já conta com alguns de seus módulos em funcionamento, que podem ser vistos na Figura 1: a coleta de dados de aeronaves, um banco de dados online com as informações coletadas e um site, onde as mesmas podem ser acessadas[1]. Dentre suas atuais

[1] Ressalta-se que o sistema possui coletores em funcionamento apenas no estado do Ceará, os quais capturam dados de uma pequena quantidade de aeronaves diariamente. O site do sistema Radar Livre disponibiliza publicamente todas as informações coletadas, e está disponível em www.radarlivre.com

propostas, destaca-se o desenvolvimento de um módulo de previsão de colisão entre aeronaves, que deu origem ao presente trabalho.

Figura 1 – Componentes do sistema Radar Livre



Fonte: Elaborada pelo autor

Em aeroportos, é comum a utilização de aves de rapina na prevenção contra conflitos entre aves e aeronaves, como explica Guedes et al. (2010). Por seu objetivo igualmente nobre, o módulo de detecção de colisão do sistema Radar Livre recebeu o apelido de CAPLAN, contração de *Caracara Plancus*, nome científico, como esclarecem Valadao, Junior e Franchin (2007), de uma ave de rapina comum no Brasil, popularmente conhecida como Carcará. Ressalta-se que ao longo do texto, o nome CAPLAN será utilizado para referir-se ao módulo citado.

Além de um algoritmo de detecção de colisão, o CAPLAN necessita de alternativas para lidar com milhares de aeronaves simultaneamente. Com foco apenas na detecção de colisões entre aeronaves, o presente trabalho nasce com o intuito de propor e avaliar o desempenho de soluções para aplicação em grande escala de um algoritmo de detecção de colisão entre aeronaves, o qual fará parte do futuro sistema de previsão de colisão do sistema de monitoramento Radar Livre.

1.1 Objetivo Geral

O objetivo deste trabalho é desenvolver e avaliar o desempenho do software de previsão de colisão entre aeronaves CAPLAN.

Por executar em uma base de dados centralizada, o algoritmo de detecção de colisão (que analisa a iminência de colisão apenas entre duas aeronaves por vez) deve ser repetido em todos os pares de aeronaves disponíveis no banco de dados do sistema. O desafio da aplicação

não reside, por tanto, na simples execução do algoritmo de detecção selecionado, mas sim em sua repetição em milhões de pares de aeronaves com resultado objetivo em tempo hábil. Esse desafio exige uma solução de escalabilidade que garanta a execução completa do algoritmo em um intervalo viável de tempo, curto o suficiente para que um alerta seja gerado a tempo de evitar uma colisão.

1.2 Objetivos específicos

Para alcançar o objetivo geral deste trabalho, os seguintes objetivos específicos foram realizados:

- Desenvolver o módulo de previsão de colisão entre aeronaves CAPLAN, usando um algoritmo de previsão selecionado em trabalhos existentes;
- Formular uma alternativa para utilização do algoritmo escolhido em grande escala, visto que o sistema irá monitorar uma grande quantidade de aeronaves simultaneamente em sua base de dados centralizada;
- Fazer uma avaliação de desempenho da solução de escalabilidade apresentada, comparando-a com a execução sem qualquer estratégia de otimização e verificando sua viabilidade para compor o módulo CAPLAN.

O primeiro destes objetivos se resume a uma pesquisa dos algoritmos de detecção de colisão propostos atualmente, na busca do mais adequado ao ambiente do sistema Radar Livre. Este sistema opera, como dito antes, com uma grande quantidade de dados combinados, o que exige um algoritmo extremamente rápido. No entanto, enquanto alguns algoritmos para este fim ganham em rapidez, os mesmos acabam perdendo em precisão por desconsiderarem algumas informações importantes, como o formato geodésico da terra. Outros, por considerarem informações como essa, acabam tornando-se complexos e lentos, embora extremamente precisos.

A precisão do algoritmo é um fator decisivo em sua velocidade de execução, o que a torna um elemento crítico do módulo CAPLAN, que portanto não pode ser desprezado. Alternativamente, podemos dar uma atenção especial ao segundo objetivo específico deste trabalho encontrando uma solução de escalabilidade bastante eficiente que compense a complexidade do algoritmo de detecção. Esta solução deve ainda garantir a execução total do algoritmo em um intervalo de tempo com consumo de memória e nível de processamento viáveis, os quais serão medidos por uma análise de desempenho que caracteriza o terceiro objetivo deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir são mostrados os principais referenciais teóricos encontrados durante as pesquisas para este trabalho.

2.1 Monitoramento Aéreo Brasileiro

Próximo aos aeroportos, os aviões são monitorados visualmente pelos controladores de voo e por radares auxiliares. Após cerca de 10 quilômetros, a aeronave passa a ser monitorada por radares de controle de aproximação (APP), que garantem uma distância mínima entre as aeronaves e previnem possíveis colisões. Fora do alcance do APP, a aeronave passa a ser monitorada pelo Departamento de Controle do Espaço Aéreo (DECEA), até chegar próximo ao seu destino (DECEA, 2015).

O DECEA tem por objetivo gerenciar as atividades no espaço aéreo brasileiro. Sua estrutura conta com quatro Centros Integrados de Defesa Aérea e Controle de Tráfego Aéreo (CINDACTA). A unidade CINDACTA I é responsável pelo espaço aéreo do Distrito Federal, Goiás, parte do Mato Grosso e Região Sudeste; a unidade CINDACTA II é responsável pela Região Sul, Mato Grosso do Sul e parte sul e oeste de São Paulo; a unidade CINDACTA III é responsável pela Região Nordeste, parte de Minas Gerais, parte do Tocantins e área oceânica que separa o Brasil da África e da Europa; e a unidade CINDACTA IV se responsabiliza pela Região Amazônica. O DECEA também conta com três subdepartamentos de supervisão, um Serviço Regional de Proteção ao Voo (SRPV), cinco Centros de Controle de Área (ACC), 47 Controles de Aproximação (APP), 59 Torres de Controle de Aeródromo (TWR), 79 Destacamentos de Controle do Espaço Aéreo (DTCEA), além das mais de 90 Estações de Telecomunicações Aeronáuticas e diversas divisões de apoio por todo o País (DECEA, 2015).

2.2 Radares

Quando estão fora do alcance dos controladores de voo, as aeronaves são monitoradas por radares. Estes aparelhos são divididos em dois tipos: os primários e os secundários. Os radares primários emitem ondas eletromagnéticas à atmosfera que retornam ao refletirem em algum obstáculo. Através da medição do tempo de ida e volta das ondas, pode-se medir a distância e posição do objeto. No entanto, radares primários não capturam dados de altitude e elevação. Os radares secundários funcionam enviando mensagens às aeronaves, que respondem

com informações da posição, velocidade, altitude, etc. Para isso, é necessário que a aeronave tenha um transreceptor (aparelho capaz de receber e enviar mensagens). Caso o avião não possua esse aparelho, o radar secundário será incapaz de encontrá-lo. Muitas aeronaves em funcionamento não possuem um transreceptor abordo e, portanto, não são identificadas por radares secundários. Assim, a maioria dos aeroportos são equipados com os dois tipos de radar (SMAAL, 2010).

Radares ainda são largamente utilizados, mas são aparelhos caros e, apesar de cumprirem bem o seu trabalho, barreiras físicas e condições atmosféricas desfavoráveis podem atrapalhar seu funcionamento. A atualização do posicionamento do avião ocorre apenas a cada 30 segundos, o que dificulta a eficiência na prevenção contra acidentes e leva à necessidade de se dispor de um método de monitoramento aéreo mais confiável e acessível. Para resolver tais problemas do sistema atual de monitoramento aéreo, foi criado o sistema CNS/ATM (Comunicação, Navegação, Vigilância e Gerenciamento de Tráfego Aéreo), que usa a tecnologia ADS-B para o compartilhamento de informações das aeronaves (GALOTTI, 1997).

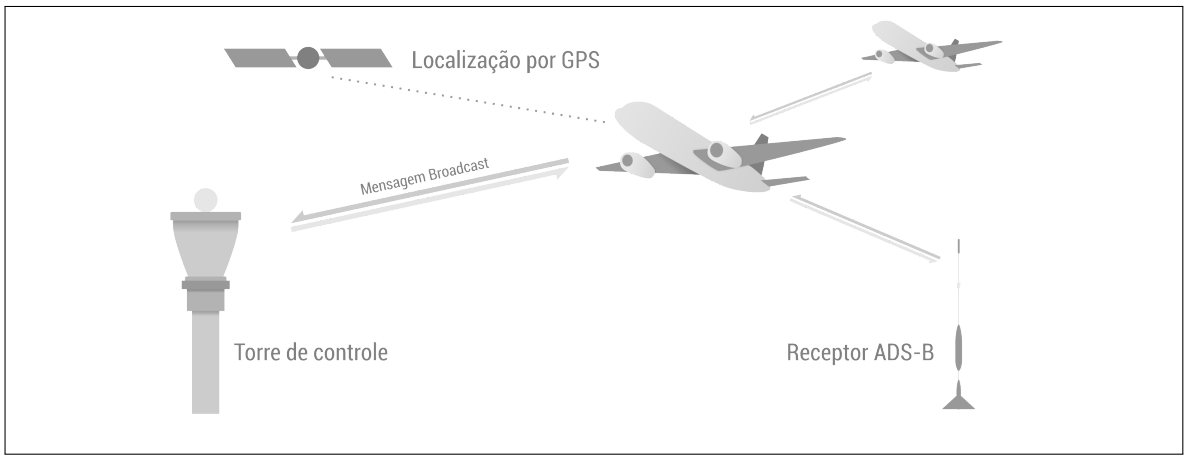
2.3 Tecnologia ADS-B

O sistema CNS/ATM pretende dar fim aos complexos e caros sistemas baseados em radares, substituindo-os pela tecnologia de Vigilância Segura Automática por Radiodifusão (ADS-B) (GALOTTI, 1997).

A tecnologia ADS-B baseia-se em sistemas modernos de geolocalização para detecção do posicionamento das aeronaves. Após serem coletados, os dados da aeronave são codificados em mensagens de 112 bits em formato hexadecimal e enviados em um intervalo configurável entre 0,5 e 2 segundos em todas as direções. Como pode ser visto na Figura 2, as mensagens podem ser coletadas e propagadas por outras aeronaves, por receptores posicionados no solo ou por torres de controle, desde que possuam um receptor ADS-B. Mesmo que uma aeronave esteja distante de qualquer torre de controle, suas mensagens podem ser propagadas por outras aeronaves até chegar em algum receptor. Dessa forma, esse formato de compartilhamento aumenta consideravelmente o alcance e a precisão do monitoramento (ZHANG; QIAO, 2008).

Apesar de ser mais barata e simples, a tecnologia ADS-B é totalmente dependente do bom funcionamento dos sistemas de geolocalização, o que os tornam elementos críticos do sistema (LESTER, 2007).

Figura 2 – Funcionamento do compartilhamento de mensagens ADS-B



Fonte: Elaborada pelo autor

2.4 O Algoritmo de Previsão de Colisão Entre Aeronaves de Gariel, Kunzi e Hansman

Com o auxílio de tecnologias modernas de coleta de dados como a ADS-B, é possível monitorar com precisão e rapidez as milhares de aeronaves que sobrevoam o planeta simultaneamente. Mas utilizar tais dados para a análise e detecção de colisão exige algoritmos em sua maioria complexos e com vários problemas de otimização. Um exemplo de algoritmo relativamente simples e eficiente é apresentado no trabalho de Gariel, Kunzi e Hansman (2011), o qual será utilizado neste trabalho em testes de escalabilidade.

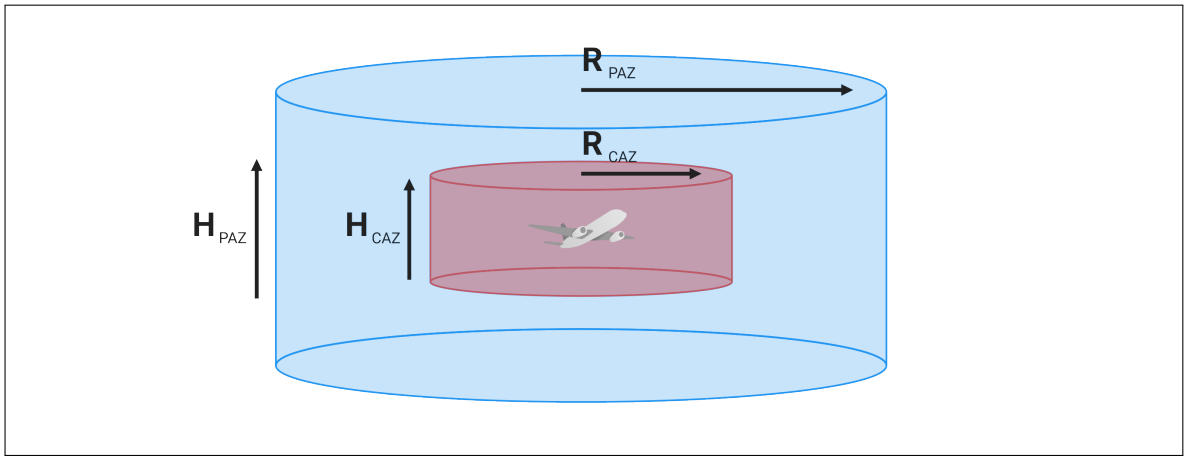
O algoritmo de Gariel, Kunzi e Hansman (2011) considera duas zonas em torno da aeronave que garantem sua segurança, como mostra a Figura 3. A primeira zona se chama Collision Airspace Zone (CAZ), uma região cilíndrica com 160 metros de raio e 60 metros de altura, cujas dimensões consideram o tamanho das aeronaves atuais e possíveis imprecisões em sua localização. Se outra aeronave cruzar essa zona, estará correndo um risco sério de colisão. A Protection Airspace Zone (PAZ) determina uma região cilíndrica em torno da aeronave, normalmente maior que a região CAZ, que não tem seu foco em prever uma colisão iminente, mas sim proporcionar um voo confortável para o piloto, mantendo-o suficientemente distante de outros voos. Seu tamanho é calculado em função de sua velocidade relativa a uma segunda aeronave. Por tanto, a região PAZ é dinâmica e varia dependendo do par de aeronaves que está sendo verificado. Como explicam Gariel, Kunzi e Hansman (2011), as dimensões da região PAZ podem ser obtidas com:

$$R_{PAZ}(t) = R_{PAZ,min} + max(0, V_R(t))\,\tau_{hor}$$

$$H_{PAZ}(t) = H_{PAZ,min} + max(0, V_{R,vert}(t))\tau_{vert}$$

onde $R_{PAZ,min} = R_{CAZ} = 160 metros$, $V_R(t)$ é a velocidade horizontal relativa entre as aeronaves no instante t, $\tau_{hor} = 10s$ é uma constante de tempo para a dimensão horizontal de PAZ, $H_{PAZ,min} = 91 metros$ é a altura mínima de PAZ, $V_{R,vert}(t)$ é a velocidade vertical relativa entre as aeronaves e $\tau_{ver} = 10s$ é uma constante de tempo para a dimensão vertical de PAZ.

Figura 3 – Regiões CAZ e PAZ em torno de uma aeronave



Fonte: Elaborada pelo autor

O algoritmo também prevê alertas distintos para cada tipo de conflito. Caso uma aeronave esteja prestes a entrar na região PAZ de outra dentro dos próximos 15 segundos, um alerta é criado, advertindo que os aviões apresentam uma distância crítica entre si. Se o conflito for detectado para os próximos 15 segundos, outra detecção PAZ é esperada antes que o alerta seja criado, evitando que alertas sejam criados quando uma aeronave cruza momentaneamente a rota de outra. Caso o conflito seja previsto para menos de 15 segundos, o alerta é criado imediatamente. O alerta CAZ funciona de forma semelhante, mas tem muito mais gravidade por indicar uma colisão iminente.

Além de verificar posição atual do avião, o algoritmo faz uma simulação da rota que será percorrida pelo mesmo nos próximos segundos, baseando-se em sua posição, velocidade e taxa de giro. O valor de t varia, portanto, de 0 a **T**, onde **T** representa o **Intervalo de Simulação**, e são verificados conflitos PAZ e CAZ para cada novo valor de posição gerado em função de t. O **Intervalo de Simulação** é um fator importante para o experimento e será detalhado na secção 4 deste documento. Na verificação de colisão entre duas aeronaves, suas respectivas rotas serão sincronizadas em relação a t e propagadas simultaneamente. Assim, caso a primeira

aeronave tenha enviado sua última informação de posição no instante $t = 1446520500$ (*timestamp* do dia 3 de novembro de 2015, às 3 horas) e a segunda tenha enviado duas informações em $t = 1446520400$ e $t = 1446520600$, será efetuado um cálculo de interpolação cúbica para a descoberta da posição da segunda aeronave exatamente no instante $t = 1446520500$. A partir dessa sincronização dos valores de t, será feita a simulação das rotas das duas aeronaves.

2.5 Análise e Detecção de Colisão em Grande Escala

Mesmo com tecnologias mais precisas para o monitoramento de aeronaves, garantir um voo seguro ainda é um grande desafio. Em seu módulo de detecção de colisão, o sistema Radar Livre pretende evitar colisões entre aeronaves utilizando as informações coletadas por seus receptores ADS-B. No entanto, se o sistema for alimentado com um grande número de aeronaves, o módulo de detecção de colisão irá lidar com um grande problema de escalabilidade. Segundo o site FlightRadar24 (2016), onde podemos acompanhar o voo de aeronaves em tempo real, às 9h30min do dia 24 de Maio de 2016 eram registradas pouco mais de 12.500 aeronaves em funcionamento ao redor do mundo. Ressalta-se que as aeronaves identificadas eram apenas aquelas que carregavam um transreceptor ADS-B e estavam no raio de alcance de alguma antena do sistema. Portanto essa quantidade deve estar bem abaixo do número real de aeronaves em pleno voo no momento da verificação. Verificar a possibilidade de colisão em tantas aeronaves de forma rápida e eficiente é o principal desafio do módulo de detecção de colisão do sistema Radar Livre.

O sistema TCAS funciona dentro da aeronave e preocupa-se em evitar colisão apenas com as aeronaves vizinhas, não necessitando de uma grande capacidade de processamento na verificação de poucas aeronaves. O sistema Radar Livre utiliza uma base de dados unificada com informações de todas as aeronaves observadas, o que obriga seu módulo de detecção de colisão a analisar a possibilidade de colisão entre todas as aeronaves registradas pelo sistema. Em um espaço aéreo pequeno cruzado por três aviões A, B e C, por exemplo, para detectar com segurança todas as possíveis colisões, seria necessário analisar um possível conflito entre os aviões A e B, entre B e C e entre A e C, totalizando 3 verificações. Em uma espaço aéreo com 12.500 aeronaves, seriam necessárias 78.118.750 verificações. Se considerarmos, apenas como suposição, que para se ter um monitoramento preciso devemos atualizar todas as previsões a cada 2 segundos, e que, ainda como suposição, em média 12.500 aeronaves sobrevoam simultaneamente a qualquer hora, chegaremos a 337.730.000.000 verificações diárias e o problema de escalabilidade fica claro.

3 TRABALHOS RELACIONADOS

Existem vários trabalhos propondo ou descrevendo algoritmos de previsão de colisão, alguns com foco em velocidade de execução, outros com foco em precisão, como os apresentados nos trabalhos de: Carbone et al. (2006), Gariel, Kunzi e Hansman (2011), Šišlák, Samek e Pěchouček (2008), Chamlou (2010) e Chamlou, Love e Moody (2008). O módulo CAPLAN exige algoritmos rápidos, capazes de analisar milhares de aeronaves em tempo hábil e, por ser esse seu foco, os algoritmos apresentados por Carbone et al. (2006) e Gariel, Kunzi e Hansman (2011) serão analisados mais detalhadamente a seguir.

Em seu trabalho, Carbone et al. (2006) apresentam um algoritmo para detecção e tratamento de colisões utilizando-se de geometria tridimensional. O mesmo é apresentado como uma alternativa aos algoritmos existentes até então, que não operavam rápido o suficiente. No método, as aeronaves são tratadas como vetores tridimensionais. Uma das aeronaves, considerada como um referencial, é o centro de uma esfera que define um volume de segurança em torno da mesma. Se uma outra aeronave cruzar essa fronteira, estará em estado de colisão iminente. Por ser simples e direto, o algoritmo pode ser executado rapidamente adequando-se a aplicações de tempo real. O método não considera, no entanto, possíveis variantes na orientação e velocidade das aeronaves, o que o torna pouco preciso.

Gariel, Kunzi e Hansman (2011) apresentam uma solução que considera variantes de orientação e velocidade da aeronave, além de protocolos que determinam o funcionamento dos alertas aos pilotos. Por considerar mais variantes, esse algoritmo é mais complexo e seu processamento é menos eficiente que o de Carbone et al. (2006). A solução de Gariel, Kunzi e Hansman (2011), no entanto, produz resultados mais precisos e portanto, foi a escolhida para este trabalho.

Os dois algoritmos focam no conflito de duas aeronaves em particular, não abordando o problema de análise de colisão entre vários aviões de forma rápida e escalável. Este trabalho se diferencia por propor e analisar o desempenho de uma solução para aplicação do algoritmo de Gariel, Kunzi e Hansman (2011) com até milhares de aeronaves simultaneamente.

4 MATERIAIS E MÉTODOS

As etapas seguidas na realização deste trabalho estão listados abaixo.

4.1 Revisão Bibliográfica

Antes de mais nada, foram realizadas pesquisas em trabalhos já existentes com foco na compreensão do funcionamento geral dos sistemas de monitoramento aéreo, bem como o conhecimento das novas tecnologias utilizadas. A tecnologia ADS-B recebeu uma atenção especial por ser um componente fundamental do sistema Radar Livre. Em seguida, foram analisados trabalhos direcionados à previsão e tratamento de colisão, alguns abordando o sistema TCAS e outros trazendo algoritmos de previsão de colisão mais modernos. Entre estes, o algoritmo apresentado no trabalho de Gariel, Kunzi e Hansman (2011) foi selecionado como o mais adequado a compor o módulo CAPLAN. As principais ferramentas de pesquisa foram os sites ACADÊMICO (2016) e IEEE (2016). As palavras-chave utilizadas na pesquisa são: monitoramento aéreo, detecção de colisão de aeronaves no espaço aéreo, TCAS e tecnologia ADS-B.

4.2 Levantamento de Requisitos Para o Módulo CAPLAN

Após as pesquisas iniciais, foi realizado um levantamento de requisitos para o desenvolvimento do módulo de previsão de colisões CAPLAN, no qual foram listadas as funcionalidades exigidas ao módulo, suas restrições de desempenho e precisão (baseadas no trabalho de Gariel, Kunzi e Hansman (2011) e nas necessidades do próprio sistema Radar Livre), e os recursos já disponíveis no sistema Radar Livre que darão suporte à previsão de colisão.

4.2.1 Requisitos Funcionais

- O módulo deve ser capaz de prever a iminência de conflito entre duas aeronaves dadas suas posições globais (latitude, longitude e altitude) e velocidade (velocidade norte-sul, velocidade leste-oeste e velocidade vertical).
- O módulo deve ser capaz de prever a iminência de conflito entre todos os pares de aeronaves presentes no banco de dados do sistema no momento de sua execução.
- O módulo deve detectar dois tipos de conflitos: o primeiro tipo indica que as aeronaves

apresentam uma distância crítica entre si e que o risco de colisão pode surgir a qualquer momento (baseado na região PAZ descrita na seção 2); o segundo tipo indica uma iminência real de colisão para os próximos segundos (baseado na região CAZ descrita na seção 2).

- Para cada conflito previsto para dentro dos próximos 15 segundos, um alerta deve ser criado imediatamente. Se um conflito for previsto para exatamente 15 segundos depois que o mesmo foi detectado, deve-se esperar uma segunda previsão de conflito para que a primeira seja validada, o que evita a criação de alarmes desnecessários quando uma aeronave cruza momentaneamente a rota de outra. Essas restrições são especificadas no trabalho de Gariel, Kunzi e Hansman (2011), e foram adotadas neste trabalho.
- Um alerta deve informar qual seu tipo, quais aeronaves estão envolvidas no conflito e quanto tempo resta até que o mesmo ocorra.
- Após sua criação, os alertas devem ser inseridos no banco de dados disponível no servidor do sistema, para que possam ser acessados pelas aplicações e mostrados aos usuários do sistema.
- O módulo deve acessar as informações das aeronaves diretamente no banco de dados disponível no servidor do sistema, de forma a obtê-los o mais rapidamente possível.
- O módulo terá disponível um servidor com 32 Giga Bytes de memória RAM e 24 núcleos de processamento, recursos que podem oscilar sua disponibilidade devido à hospedagem do servidor Web do sistema e outros processos relativos ao sistema operacional.

4.2.2 Requisitos Não Funcionais

- O módulo deve executar toda a previsão de colisão em tempo de gerar todos os alertas com no mínimo 15 segundos de antecedência.
- O módulo deve executar toda a previsão de colisão consumindo o mínimo de memória RAM possível, com um máximo de 30 Gigabytes.

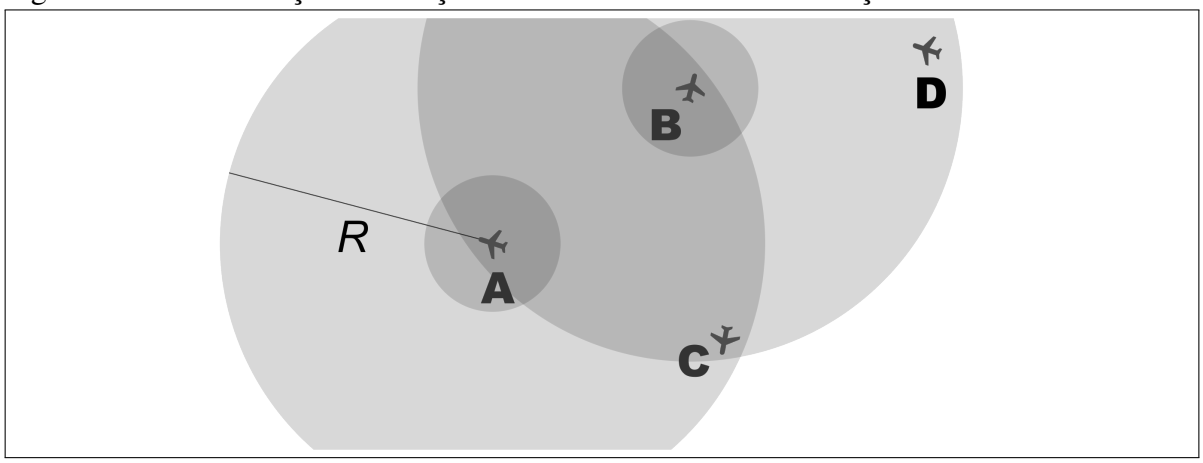
4.3 Desenvolvimento do módulo CAPLAN

Devido à centralização de dados, todas as informações do sistema são processadas em um único lugar, o que exige uma grande capacidade de memória e de processamento no sistema. O módulo de detecção de colisão CAPLAN deve, portanto, otimizar ao máximo seu tempo de execução e uso de memória.

Baseadas no funcionamento do algoritmo de detecção de colisão e nos recursos disponíveis no ambiente de execução, podem ser aplicadas algumas técnicas de otimização para proporcionar um melhor desempenho ao módulo CAPLAN, tornando-o viável ao sistema Radar Livre. O servidor onde o módulo funcionará dispõe de 32 Gigabytes de memória RAM, uma quantidade considerável que dá suporte à técnica **Memorização**, normalmente usada, como explica Chandran e Grandoni (2005), em algoritmos recursivos de complexidade exponencial, onde um mesmo passo é reexecutado várias vezes. A técnica armazena os resultados que seriam repetidos em uma estrutura para que sejam acessados durante a execução do algoritmo, impedindo que sejam calculados mais de uma vez. Em seu algoritmo de previsão de colisão, o módulo CAPLAN faz uma simulação da rota a ser percorrida pela aeronave nos próximos segundos, repetindo-a quando a aeronave é verificada em relação às demais. A simulação é, no entanto, baseada em um cartesiano local, relativo a uma das aeronaves, o que produz valores diferentes para cada simulação. Isso mostra que, nesse caso, a técnica de **Memorização** não é adequada ao módulo. Outras técnicas, porém, se mostraram mais promissoras, e estão listadas a seguir.

- **Primeira Estratégia: Raio Mínimo de Verificação.** Um Raio Mínimo de Verificação consiste em uma distância mínima que duas aeronaves devem apresentar entre si para que o algoritmo de detecção possa ser executado em ambas. A região de segurança PAZ, apresentada no trabalho de Gariel, Kunzi e Hansman (2011) funciona como um raio mínimo de verificação para as aeronaves. No exemplo da Figura 4, a aeronave *D* está fora do raio de verificação de *A* e, portanto, o algoritmo de detecção de colisão não analisará *A* em relação a *D*. Assim, esta solução busca evitar a execução do algoritmo em aeronaves demasiadamente distantes, poupando processamento, memória e tempo de execução.

Figura 4 – Demonstração da solução Raio Mínimo com Memorização



Fonte: Elaborada pelo autor

O próprio algoritmo de detecção de Gariel, Kunzi e Hansman (2011), proporciona o suporte à técnica **Raio Mínimo de Verificação** com sua região PAZ. Porém, como o algoritmo simula a rota a ser percorrida por cada aeronave nos próximos 15 segundos (intervalo mínimo para o lançamento de alarmes), a região de Raio Mínimo de Verificação será obtida pela região PAZ estendida com a distância que seria percorrida pela aeronave no intervalo de 15 segundos. A não intersecção das regiões de Raio Mínimo de Verificação de duas aeronaves implica que nenhum alarme será lançado agora, e a custosa simulação de rota a ser percorrida pelas aeronaves não será executada.

- **Segunda Estratégia: Multiprocessamento Simétrico.** O multiprocessamento simétrico é, segundo Arimilli et al. (2004), uma forma de dividir a execução de um programa entre vários processadores compartilhando a mesma memória. Essa divisão é feita geralmente com o uso de *threads*, ou linhas de execução, onde o processo principal da aplicação se divide em 2 ou mais subprocessos que são executados em paralelo (BLUMOFE et al., 1996). Isso proporciona um aumento linear na eficiência do processamento, proporcional à quantidade de processadores disponíveis. O computador disponível para este experimento tem 24 núcleos de processamento, nos quais os pares de aeronaves podem ser divididos e verificados paralelamente.

  O problema de escalabilidade consiste em uma combinação de pares de aeronaves, nos quais o algoritmo de detecção será aplicado. Durante sua execução, o módulo gera uma lista com todos os pares de aeronaves possíveis. Com a técnica de **Multiprocessamento Simétrico**, a lista de pares é dividida pelo número de processadores disponíveis e a mesma quantidade de *threads* é criada. Cada *thread* executará uma parte da lista

independentemente das demais, e os alarmes serão gerados em tempo de execução de forma assíncrona.

- **Terceira Estratégia: Divisão do Espaço Aéreo em Áreas.** Assim como a estratégia de **Raio Mínimo de Verificação**, a **Divisão do Espaço Aéreo em Áreas** considera que aeronaves muito distantes não irão colidir num curto intervalo de tempo e o algoritmo pode simplesmente ignorá-las e analisar somente aeronaves com real chance de colisão a curto prazo, mas, o objetivo agora é dividir o espaço aéreo global em áreas e analisar apenas aeronaves em regiões próximas, evitando analisar aeronaves muito distantes e poupando processamento.

  Existem muitas formas de dividir o globo terrestre em áreas. Estas podem ser representadas, por exemplo, como regiões triangulares que formam um sólido geométrico de 4 faces (tetraedro), 8 faces (octaedro), ou até mesmo 20 faces (icosaedro). Porém, tais representações tridimensionais são computacionalmente complexas, o que as afasta do seu objetivo inicial de proverem uma otimização referente ao tempo de execução do algoritmo. Uma representação mais simples, é a divisão do globo terrestre em fatias delimitadas por latitudes específicas. Nesse caso, as aeronaves podem ser classificadas nas fatias com uma simples verificação de sua latitude e apenas aeronaves na mesma fatia ou em fatias vizinhas serão verificadas entre si pelo algoritmo, poupando memória, processamento e tempo de execução. Devido a esses benefícios, esta foi escolhida e implementada neste experimento.

O módulo foi implementado na linguagem de programação C++, devido, como afirmam Gherardi, Brugali e Comotti (2012), a sua alta velocidade de execução e flexibilidade no tratamento de memória, sob o paradigma de orientação a objetos. Neste paradigma, a **classe** é uma parte do código responsável por descrever atributos e comportamentos, os quais são utilizados durante a execução do programa na construção de um **objeto**, um componente que, combinado com outros, compõe um software em execução. Cada **classe** pode ser usada como base na implementação de outras mais complexas, mas que conservam seus atributos, o que, no paradigma de orientação a objetos, é chamado de **herança de classe**. As **classes** usadas no módulo CAPLAN são listadas a seguir:

- **Math**: responsável pelos cálculos vetoriais e geoespaciais presentes no algoritmo de previsão de colisão, incluindo multiplicação, adição, subtração, normalização, rotação e translação de vetores, o cálculo da distância entre dois pontos na superfície da terra com

a **fórmula de Haversine** (ROBUSTO, 1957) e um método de interpolação cúbica com a *Spline Catmull-Rom* (DEROSE; BARSKY, 1988) utilizado na sincronização das rotas das aeronaves durante a execução do algoritmo.

- **Combinator**: responsável por formar os pares de aeronaves a serem processados pelo algoritmo e dividi-los em grupos de forma a paralelizar sua execução com a estratégia de **Multiprocessamento Simétrico**.
- **CollisionDetector**: responsável por aplicar o algoritmo de detecção de colisão a cada par de aeronaves, aplicando as estratégias de **Divisão do Espaço Aéreo em Áreas** e **Raio Mínimo de Verificação** e gerando os alertas para o sistema.
- **Repository**: responsável por carregar as informações das aeronaves. Essa **classe** é a base da **classe RealRepository**, que obtém as informações diretamente do banco de dados.

Devido à necessidade da realização de testes, o módulo utiliza recursos da linguagem C++ e do próprio sistema operacional, no caso deste experimento o **Ubuntu 16.04**[1], uma distribuição **Linux**[2], para obter informações sobre consumo de memória e tempo de execução, que são armazenadas em arquivos para posterior análise. Entrementes, a **classe SinteticRepository** substituiu o **classe RealRepository** na obtenção de informações de aeronaves. Isso se deve a sua capacidade de disponibilizar dados sintéticos sob as especificações exigidas pelos testes. Se um destes pretende analisar o módulo buscando conflitos em 20.000 aeronaves simultaneamente, por exemplo, a classe **SinteticRepository** irá sinterizar 20.000 aeronaves, gerando randomicamente suas respectivas latitudes, longitudes, altitudes e velocidades, respeitando intervalos encontrados em dados reais.

Ainda para automação dos testes, foi criado um *script* na linguagem de programação Shell Bash, que executa os testes e gera uma planilha com os resultados, que pode ser encontrada na seção 5 deste trabalho.

4.4 Realização da Análise de Desempenho

Com o módulo em funcionamento e o suporte a testes preparado, as etapas da análise de desempenho tomaram o foco, as quais são listadas a seguir:

[1] Mais informações em https://www.ubuntu.com/
[2] Mais informações em https://www.linux.com/

4.4.1 Conhecer o sistema

O módulo CAPLAN funcionará em uma máquina com 24 núcleos de processamento e 32 Giga Bytes de Memória Ram. Ao seu lado estará funcionando o servidor Web do sistema Radar Livre, que é implementado na linguagem Python em sua versão 2.7.12 e usa o framework DJango[3] em sua versão 1.9. O servidor provê uma interface de acesso (API) que pode ser usada pelos coletores para envio de novas informações, e por suas aplicações para acesso aos dados registradas pelo sistema. O servidor conta com o Sistema de Gerenciamento de Banco de Dados PostgreSQL[4], uma poderosa e robusta ferramenta de código aberto onde estão armazenados todos os dados coletados, e que será acessado pelo módulo.

4.4.2 Escolher as Métricas

As métricas do experimento são valores variáveis, medidos durante a avaliação de desempenho, que descrevem, nesse caso, os recursos consumidos pelo software, tais como:

- **Consumo de memória**: a quantidade de memória RAM consumida pelo módulo CAPLAN não deve ultrapassar o limite de 30 Giga Bytes, visto que a quantidade total disponível no servidor é de 32 Giga Bytes e além do módulo, estará em execução o servidor Web do sistema.
- **Tempo de execução**: o tempo de execução do algoritmo de detecção de colisão em todas as aeronaves deve ser curto o suficiente para que qualquer alarme seja emitido 15 segundos antes do possível conflito, seja ele PAZ ou CAZ.

4.4.3 Definir Fatores e Níveis

Os fatores que podem afetar os resultados do experimento, principalmente em relação a tempo de execução e consumo de memória, e seus respectivos níveis são apresentados na Tabela 1:

Na Tabela 1, SO indica sem otimização, DA indica **Divisão em Áreas**, MPS indica **Multiprocessamento Simétrico** e DA+MPS indica a junção das duas técnicas. Ressalta-se que a técnica de **Raio mínimo de Verificação** já é implementada no algoritmo de Gariel, Kunzi e Hansman (2011) e é executada em todos os experimentos.

[3] Mais informações em www.djangoproject.com
[4] Mais informações em www.postgresql.org/about

Tabela 1 – Fatores e níveis.

Fator	Níveis
Número de *threads* de execução	n, $n-2$ e $n-4$, onde n representa o número de processadores disponíveis
Intervalo de simulação	20, 30, 40, 50 e 60 segundos
Intervalo entre áreas	1º, 2º, 3º, 4º e 5º
Quantidade de aeronaves sintetizadas	500, 1000, 5000, 10000, 15000, 20000, 25000 e 30000
Técnicas de otimização	SO, DA, MPS, DA+MPS

Fonte: Produzido pelos autores

- **Número de *threads* de execução**: a execução do algoritmo será paralelizada em uma quantidade configurável de *threads*. Para este experimento, serão consideradas as quantidades n, $n-2$ e $n-4$, onde n representa o número de núcleos de processamento disponíveis no computador de testes, no caso deste trabalho $n = 24$. No caso de $n-2$ e $n-4$, as quantidades subtraídas são compensações às *threads* utilizadas no restante do módulo de detecção (a *thread main*, a *thread* responsável pela execução da combinação, a *thread* responsável pelo acesso ao banco de dados e a *thread* responsável pelo processamentos dos alarmes gerados). Essa compensação garante que o número de *threads* seja o mesmo que o número de núcleos de processamento. Porém, como algumas partes do módulo exigem mais poder de processamento que outras, não há garantia de que uma mesma quantidade de *threads* e núcleos proporcione um aproveitamento máximo do paralelismo, algo que apenas experimentos podem mostrar.
- **Intervalo de simulação**: durante a execução do algoritmo de detecção de colisão, este realiza uma simulação da rota a ser percorrida por cada aeronave em um intervalo configurável de tempo. Este intervalo interfere diretamente no tempo de processamento do algoritmo e a diferença entre os mesmos (intervalo de simulação e tempo de processamento) deve ser de no mínimo 15 segundos, de forma que qualquer alarme seja emitido 15 segundos antes do possível conflito. Os intervalos testados serão de 20, 30, 40, 50 e 60 segundos.
- **Intervalo entre áreas**: com a estratégia de divisão em áreas, o globo terrestre é dividido em fatias de intervalo de latitude configuráveis. Se o intervalo for de 1 grau, por exemplo, as fatias terão altura de um grau de latitude. Para este experimento, serão testados intervalos de 1 a 5 graus.

4.4.4 Escolher a Técnica de Avaliação

Neste trabalho foi utilizada a técnica **Trace-based Simulation**, que é amplamente utilizada na Ciência da Computação para fazer simulações e observações durante a execução de um programa e assim medir os recursos gastos, procurar possíveis falhas e avaliar seu desempenho (OWEZARSKI; LARRIEU, 2004).

4.4.5 Executar a Avaliação e Obter os Resultados

O módulo CAPLAN foi executado várias vezes por um *script* no servidor do sistema Radar Livre. Em sua versão de teste o módulo recebe parâmetros de execução que especificam a quantidade de aeronaves a ser sintetizada, quais técnicas de otimização devem ser utilizadas e quantas vezes o teste deve ser repetido. Os valores utilizados para a quantidade de aeronaves foram 100, 500, 1000, 5000, 10000, 15000, 20000, 25000 e 30000. Para cada um destes valores, foram testadas as seguintes possibilidades de execução: sem técnicas de otimização, apenas com a técnica **MPS**, apenas com a técnica **DA** e com as duas técnicas simultaneamente. A técnica de **Raio Mínimo de Verificação** participou de todos os experimentos pois já fazia parte do algoritmo de detecção que é especificada no trabalho de Gariel, Kunzi e Hansman (2011), onde a região de segurança PAZ representa o raio mínimo. Além disso, cada experimento foi repetido 5 vezes e os valores finais do tempo de execução e consumo de memória de cada experimento são obtidos pela média aritmética dos valores obtidos pelas repetições dos mesmos. No total, 180 testes foram executados.

5 RESULTADOS

A seguir são mostrados os resultados obtidos pela realização da análise de desempenho.

5.1 Fatores e os Níveis Mais Promissores

Para os fatores citados anteriormente, foram testadas os vários níveis apresentados na seção anterior em busca dos que garantissem melhor desempenho à execução do módulo. Os níveis que apresentaram melhores resultados são apresentados na Tabela 2. Os fatores quantidade de aeronaves sintetizadas e técnicas de otimização não foram reduzidos justamente para serem comparados.

Tabela 2 – Fatores e níveis mais favoráveis.

Fator	Níveis com melhor resultado
Número de *threads* de execução	$n-2$, onde n representa o número de processadores disponíveis
Intervalo de simulação	60 segundos
Intervalo entre áreas	1^{o}
Técnicas de otimização	DA+MPS

Fonte: Produzido pelos autores

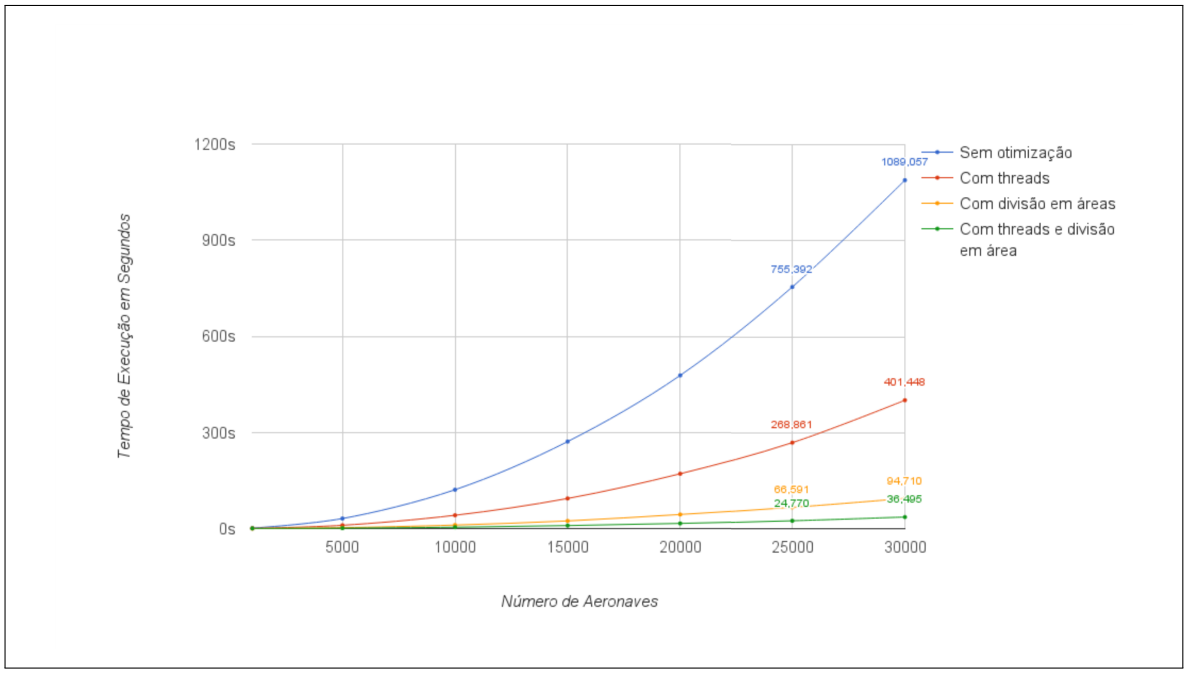
Como esperado, a utilização de $n-2$ *threads*, e não n ou $n-4$, mostrou melhores resultados em relação ao tempo de execução, pois neste caso dois núcleos de processamento são reservados para execução das *threads* referentes as partes do módulo que não executam o algoritmo, as quais não exigem um grande poder processamento. Já os testes com o **Intervalo de simulação** surpreenderam ao mostrar que o melhor nível dentre os testados foi o de 60 segundos, devido ao aumento da complexidade do algoritmo proporcional ao aumento deste nível.

O resultado mais previsível foi obtido com variação nos níveis de **Intervalos entre áreas**. Quanto menor o intervalo, menor a quantidade de pares de aeronaves formados e assim, menor o tempo de execução. No entanto, um valor muito pequeno faria aeronaves próximas serem classificadas em áreas diferentes, o que mostra que deve haver um limite inferior para esse nível. A descoberta desse limite exige complexos testes que envolvem, por exemplo, a velocidade máxima atingida por uma aeronave, que determina o quão grande deve ser o **Intervalo entre áreas** para que a aeronave não percorra esta distância em menos de 15 segundos. Esses testes, no entanto, não fazem parte deste experimento, e o valor de 1^{o} foi adotado como melhor resultado.

5.2 Análise dos resultados obtidos

Os resultados de todos os testes do módulo CAPLAN foram armazenados automaticamente pelo *script* de execução e formatados como texto, planilhas e gráficos. Esses formatos foram utilizados para uma visualização mais rica dos valores e os gráficos gerados estão apresentados nas Figuras 5 e 6.

Figura 5 – Resultado dos testes com o módulo CAPLAN em relação ao tempo de execução.

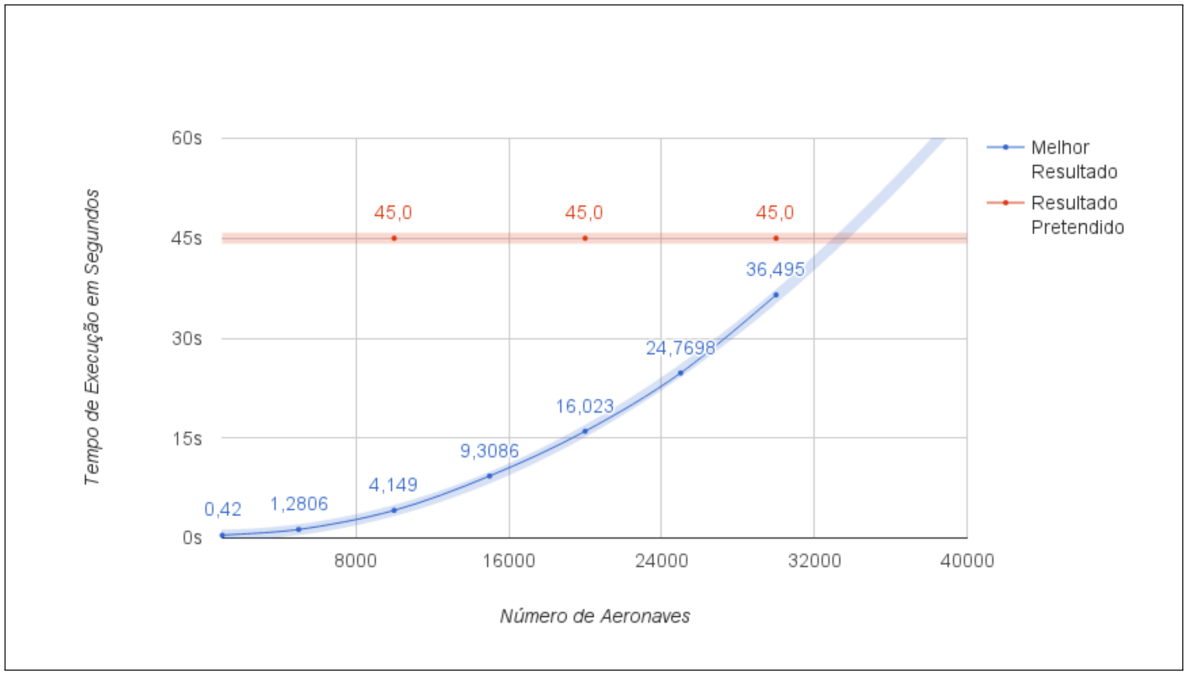


Fonte: Elaborada pelo autor

Na Figura 5, podem ser vistos os valores de tempo de execução para algumas quantidades de aeronaves analisadas. A partir de 5000 aeronaves os valores ganham uma razão mais constante entre si. Após este ponto, a técnica de **MPS** utiliza 40% do tempo de execução da abordagem não otimizada, a técnica de **Divisão em Áreas** utiliza 9% e as duas técnicas juntas utilizam apenas 3,5%, o que representa uma redução de 900 segundos na abordagem menos otimizada para 30 segundos na mais otimizada no teste com 30000 aeronaves. Mesmo com a clara diminuição no tempo pelas estratégias de otimização, o módulo deve ainda ser capaz de emitir alertas 15 segundos antes da ocorrência do conflito. Os intervalos de confiança, calculados para um nível de confiança de 99%, foram na grande maioria dos casos muito pequenos para serem mostrados na figura e portanto estão omitidos (a partir de 5000 aeronaves, não há qualquer interseção entre intervalos de cenários diferentes para um mesmo número de aeronaves).

Na Figura 6, podemos analisar o tempo de execução das estratégias de otimização com o resultado pretendido.

Figura 6 – Comparação entre o resultado pretendido e o obtido



Fonte: Elaborada pelo autor

Com uma propagação de 60 segundos, o módulo CAPLAN deve executar sua verificação de colisão em no máximo 45 segundos, permitindo que os alertas sejam criados com 15 segundos de antecedência. Segundo o gráfico da Figura 6, o tempo de execução do módulo ultrapassará o valor pretendido de 45 segundos com aproximadamente 33000 aeronaves, o que torna este o limite operacional do módulo em relação à quantidade de aeronaves. É importante ressaltar que o gráfico representa o pior caso, isto é, o tempo necessário para avaliar todas as possíveis colisões iminentes. Entretanto, ao detectar uma colisão iminente o alerta deve ser lançado imediatamente, o que significa que o tempo médio será bem melhor que o pior caso, mas o sistema deve basear-se no pior caso para garantir que todas as possíveis colisões sejam detectadas a tempo.

O consumo de memória RAM chegou ao máximo de 2,5 Gigabytes com 30000 aeronaves, mostrando que o tempo de execução ainda é o principal obstáculo do sistema.

6 CONSIDERAÇÕES FINAIS

Após a realização e interpretação de todos os testes, os resultados são bastante promissores. A quantidade de memória necessária para execução do módulo é inferior a 3 Gigabytes, bem abaixo dos 32 Gigabytes disponíveis no servidor do sistema. O tempo de execução se mostrou viável até uma quantidade de 33000 aeronaves, um resultado bastante satisfatório para a primeira versão do módulo CAPLAN. O sistema não deverá cobrir, inicialmente, uma região maior que o estado do Ceará, mas é capaz de lidar com a atual quantidade de aeronaves ativas simultaneamente usando ADS-B em todo o mundo.

Resta, porém, a tarefa de integração do recém criado módulo de previsão de colisão ao sistema Radar Livre, pois os dados utilizados até agora foram sintetizados, além da resolução de alguns problemas que vieram à tona com o experimento, como o fato de aeronaves em aeroportos lançarem alertas falsos por apresentarem uma pequena distância entre si.

Além de mostrar a viabilidade do CAPLAN, esta monografia destaca o trabalho do grupo de pesquisa responsável pelo desenvolvimento do sistema Radar Livre, uma iniciativa que faz o Brasil acompanhar os avanços tecnológicos mais recentes em monitoramento aéreo.

REFERÊNCIAS

ACADÊMICO, G. **Google Acadêmico**. 2016. <https://scholar.google.com/>.

ACRO. **Escritório de Registros de Acidentes Aéreos**. 2016. <http://www.baaa-acro.com/>.

ARIMILLI, R. K.; GUTHRIE, G. L.; STARKE, W. J.; WILLIAMS, D. E. **High performance symmetric multiprocessing systems via super-coherent data mechanisms**. [S.l.]: Google Patents, 2004. US Patent 6,785,774.

BLUMOFE, R. D.; JOERG, C. F.; KUSZMAUL, B. C.; LEISERSON, C. E.; RANDALL, K. H.; ZHOU, Y. Cilk: An efficient multithreaded runtime system. **Journal of parallel and distributed computing**, Elsevier, v. 37, n. 1, p. 55–69, 1996.

CARBONE, C.; CINIGLIO, U.; CORRARO, F.; LUONGO, S. A novel 3d geometric algorithm for aircraft autonomous collision avoidance. **Proceedings of the 45th IEEE Conference on Decision and Control**, IEEE, San Diego, CA, p. 1580 – 1585, dez. 2006. ISSN 0191-2216. Acessado em: 10 abr. 2016. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4177282>.

CHAMLOU, R. Design principles and algorithm development for two types of nextgen airborne conflict detection and collision avoidance. In: IEEE. **Integrated Communications Navigation and Surveillance Conference (ICNS), 2010**. [S.l.], 2010. p. N7–1.

CHAMLOU, R.; LOVE, W. D.; MOODY, C. Exploration of new algorithms for airborne collision detection and avoidance to meet nextgen capabilities. **2008 IEEE/AIAA 27th Digital Avionics Systems Conference**, IEEE, St. Paul, MN, p. 2.D.5–1 – 2.D.5–13, out. 2008. ISSN 2155-7195. Acessado em: 3 abr. 2016. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4702789>.

CHANDRAN, L. S.; GRANDONI, F. Refined memorization for vertex cover. **Information Processing Letters**, Elsevier, v. 93, n. 3, p. 125–131, 2005.

DECEA. **Departamento de Controle do Espaço Aéreo**. 2015. Página WEB. Acessado em: 11 mai. 2016. Disponível em: <www.decea.gov.br>.

DEROSE, T. D.; BARSKY, B. A. Geometric continuity, shape parameters, and geometric constructions for catmull-rom splines. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 7, n. 1, p. 1–41, jan. 1988. ISSN 0730-0301. Disponível em: <http://doi.acm.org/10.1145/42188.42265>.

FAA. **Office of NextGen**. 2015. Página WEB. Acessado em: 20 mai. 2016. Disponível em: <www.faa.gov/about/office_org/headquarters_offices/ang>.

FLIGHTRADAR24. **Tráfego Aéreo em Tempo Real**. 2016. <https://www.flightradar24.com/>.

GALOTTI, V. P. **The Future Air Navigation System (FANS)**. [S.l.: s.n.], 1997.

GARIEL, M.; KUNZI, F.; HANSMAN, R. J. An algorithm for conflict detection in dense traffic using ads-b. **Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th**, IEEE, Seattle, WA, p. 4E3–1 – 4E3–12, out. 2011. ISSN 2155-7195. Acessado em: 10 abr. 2016. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6095916>.

GHERARDI, L.; BRUGALI, D.; COMOTTI, D. A java vs. c++ performance evaluation: a 3d modeling benchmark. In: SPRINGER. **International Conference on Simulation, Modeling, and Programming for Autonomous Robots**. [S.l.], 2012. p. 161–172.

GUEDES, F. L.; BRAND, D. H.; LINHARES, B. d. P.; PAIVA, L. V. de. Avifauna relacionada ao risco de colisões aéreas no aeroporto internacional presidente juscelino kubitschek, brasília, distrito federal, brasil. **Conexão SIPAER**, v. 2, n. 1, p. 230–243, 2010.

IEEE. **O Instituto de Engenheiros Eletricistas e Eletrônicos**. 2016. <https://www.ieee.org/>.

KOCHENDERFER, M. J.; EDWARDS, M. W. M.; ESPINDLE, L. P.; KUCHAR, J. K.; GRIFFITH, J. D. Airspace encounter models for estimating collision risk. **Journal of Guidance, Control, and Dynamics**, v. 33, n. 2, p. 487–499, 2010.

LEE, H.-C. Implementation of collision avoidance system using tcas ii to uavs. **IEEE Aerospace and Electronic Systems Magazine**, v. 21, n. 7, p. 8–13, July 2006. ISSN 0885-8985.

LESTER, E. A. **Benefits and incentives for ADS-B equipage in the national airspace system**. Tese (Doutorado) — Massachusetts Institute of Technology, 2007.

OLIVEIRA, A. V.; SALGADO, L. Reforma regulatória e bem-estar no transporte aéreo brasileiro: e se a flexibilização dos anos 1990 não tivesse ocorrido. **Documento de Trabalho N. 013–Acervo Científico do Núcleo de Estudos em Competição e Regulação do Transporte Aéreo (NECTAR)**, 2006.

OWEZARSKI, P.; LARRIEU, N. A trace based method for realistic simulation. In: **Communications, 2004 IEEE International Conference on**. [S.l.: s.n.], 2004. v. 4, p. 2236–2239 Vol.4.

ROBUSTO, C. C. The cosine-haversine formula. **The American Mathematical Monthly**, Mathematical Association of America, v. 64, n. 1, p. 38–40, 1957. ISSN 00029890, 19300972. Disponível em: <http://www.jstor.org/stable/2309088>.

ROJAS, R. G.; CHEN, Y. C. Improved computer simulation of the tcas iii circular array mounted on an aircraft. In: **Antennas and Propagation Society International Symposium, 1989. AP-S. Digest**. [S.l.: s.n.], 1989. p. 1200–1203 vol.3.

ŠIŠLÁK, D.; SAMEK, J.; PĚCHOUČEK, M. Decentralized algorithms for collision avoidance in airspace. In: INTERNATIONAL FOUNDATION FOR AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS. **Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2**. [S.l.], 2008. p. 543–550.

SMAAL, B. **Tecnologia no controle de tráfego aéreo**. 2010. Página WEB. Acessado em: 20 mai. 2016. Disponível em: <http://www.tecmundo.com.br/3908-tecnologia-no-controle-de-trafego-aereo.htm>.

VALADAO, R. M.; JUNIOR, O. M.; FRANCHIN, A. G. A avifauna no parque municipal santa luzia, zona urbana de uberlândia, minas gerais. **Bioscience Journal**, v. 22, n. 2, 2007.

WILLIAMSON, T.; SPENCER, N. A. Development and operation of the traffic alert and collision avoidance system (tcas). **Proceedings of the IEEE**, IEEE, v. 77, p. 1735 – 1744, nov. 1989. ISSN 0018-9219. Acessado em: 10 mai. 2016. Disponível em: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=47735>.

ZEITLIN, A. D.; LOVE, W. D.; CIEPLAK, J. J. Enhancements to the next generation collision avoidance system: opportunities for greater safety and efficiency. In: **Digital Avionics Systems Conference, 1995., 14th DASC**. [S.l.: s.n.], 1995. p. 146–151.

ZHANG, Y.; QIAO, J. **ADS-B radar system**. [S.l.]: Google Patents, 2008. US Patent 7,414,567.