Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Um Algoritmo Genético Paralelo para o problema integrado de Determinação de Frequência de Voos e Alocação de Frota

Nicolas Soares Martins

JUIZ DE FORA
AGOSTO, 2025

Um Algoritmo Genético Paralelo para o problema integrado de Determinação de Frequência de Voos e Alocação de Frota

Nicolas Soares Martins

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Heder Soares Bernardino

JUIZ DE FORA

AGOSTO, 2025

Um Algoritmo Genético Paralelo para o problema integrado de Determinação de Frequência de Voos e Alocação de Frota

Nicolas Soares Martins

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

Aprovada por:

Heder Soares Bernardino
Doutor

Luciana Brugiolo Gonçalves
Doutora

José Eduardo Henriques da Silva
Doutor

JUIZ DE FORA
15 DE AGOSTO, 2025

Aos meus amigos e família.

Aos pais, pelo apoio e sustento.

Resumo

O setor aéreo de transporte, tanto de carga como de passageiros, possui crescente demanda, adquirindo cada vez mais espaço na economia brasileira. Para tanto, a velocidade da criação de rotas, escalonamento de tripulação e alocação de aeronaves são tópicos que requerem um planejamento como um todo, focando em solucionar cada um dos problemas em seu domínio para compor o plano final de operação, sendo estes os problemas de Alocação de Frota (FAP), o Roteamento de Aviões (ARP), a Determinação de Frequência de Voos (FAS) e o Escalonamento de Tripulação (CSM, geralmente dividido em subproblemas como o Pareamento de Tripulação e o *Crew Rostering Problem*). Tradicionalmente, estes problemas são modelados utilizando técnicas de programação linear inteira mista, com o uso de resolvedores para solução do modelo. Algoritmos Genéticos (AGs), apesar de serem uma alternativa quanto ao uso de resolvedores para a resolução destes problemas, podem possuir custo operacional proibitivo. A implementação de computação de alto desempenho, como a programação paralela, é uma forma de mitigar este alto custo computacional. Aqui, é proposto um algoritmo genético modificado para lidar com um problema integrado, no qual soluções candidatas para um ou mais problemas são mapeadas em uma mesma solução. Especificamente, este algoritmo se propõe a solucionar os problemas FAP e FAS. Para a avaliação deste algoritmo, foram realizados dois experimentos em sete cenários de simulação com orçamento computacional distinto, comparando os resultados obtidos em termos de qualidade e tempo de execução com a literatura, demonstrando o ganho de velocidade para obter uma solução nos três primeiros cenários simulados.

Palavras-chave: planejamento aéreo, determinação de frequência, escalonamento de tripulação, determinação de frota, otimização.

Abstract

The air transport sector, both for cargo and passengers, has growing demand, increasingly gaining space in the Brazilian economy. To this end, the speed of route creation, crew scheduling, and aircraft allocation are topics that require overall planning, focusing on solving each of the problems in their domain to compose the final operation plan. These include the Fleet Allocation Problem (FAP), the Aircraft Routing Problem (ARP), the Frequency Assignment Problem (FAS), and the Crew Scheduling Problem (CSM, divided into sub-problems such as Crew Pairing and the Crew Rostering Problem). Traditionally, these problems are modeled using mixed-integer linear programming techniques, with the use of solvers to solve the model. Although Genetic Algorithms (GAs) are an alternative to using solvers for solving these problems, they can have prohibitive operational costs. The implementation of high-performance computing, such as parallel programming, is a way to mitigate this high computational cost. Here, a modified genetic algorithm is proposed to deal with an integrated problem, in which candidate solutions for one or more problems are mapped into the same solution. Specifically, this algorithm aims to solve the FAP and FAS problems. For the evaluation of this algorithm, two experiments were conducted in seven simulation scenarios with distinct computational budgets, comparing the results obtained in terms of quality and execution time with the literature, demonstrating the speed improvements to obtain a solution in the first three simulated scenarios.

Keywords: airline scheduling, crew scheduling, frequency assignment, fleet assignment, optimization.

Agradecimentos

A todos os meus parentes, pelo encorajamento e apoio.

Ao professor Heder pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Do not feel bad about it. We are alive, after all. And being alive is pretty much a constant stream of embarrassment."

POD-153 (Nier Automata)

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
ICE	Instituto de Ciências Exatas
FAP	Fleet Assignment Problem
FAS	Frequency Assignment Problem
ARP	Aircraft Routing Problem
CSP	Crew Scheduling Problem
AG	Algoritmo Genético
MSGA	Master-Slave Genetic Algorithm
CUDA	Compute Unified Device Architecture
GPU	Graphics Processing Unit

1 Introdução

O setor aéreo retomou o crescimento pós-pandemia no mundo todo (DEGASPERI, 2024), podendo-se observar um aumento na demanda para o uso em transporte de cargas e passageiros. No Brasil, durante o ano de 2024, o setor aéreo movimentou quase 118 milhões de passageiros, sendo este o segundo maior desempenho histórico (MAZZA, 2025), além de ter contribuído com mais de 81 bilhões de reais para o PIB do país em 2023 (MAZZA, 2024). Portanto, o planejamento de companhias aéreas, tanto regionais quanto globais, desempenha papel crucial para suprir essa nova demanda, carecendo de ferramentas e métodos para providenciar em tempo hábil os artefatos necessários para a operação.

Dos problemas enfrentados pelo setor, a Alocação de Frota (FAP) e a Determinação de Frequência de Voos (FAS) são dois problemas distintos, porém relacionados, uma vez que a escolha da frota afeta a frequência de voos e vice-versa. O FAP tem como foco determinar quais tipos de aeronaves devem ser designadas para cada rota. O FAS tem como objetivo determinar a frequência de voos, isto é, a quantidade de voos realizados. Ambos os problemas possuem como foco a minimização dos custos e o aumento da eficiência de operação, maximizando assim o somatório de lucro dos voos. Devido à maneira como cada subproblema afeta significativamente a operação do setor, a solução de um subproblema individualmente pode afetar a solução global, podendo gerar uma solução inválida, considerando que geralmente esses subproblemas são resolvidos sequencialmente.

Para tanto, o estudo da solução de problemas integrados é observado atualmente, como no trabalho de Jesus, Nagano e Celestino (2023), focando em uma solução que lide com diversos dos subproblemas ao mesmo tempo. Atualmente, é utilizada a formulação como um problema de programação linear (ou não linear) de inteiros (RUTHER et al., 2017), geralmente implementando esses modelos utilizando resolvedores comerciais, computação evolucionista (DEVECI; DEMIREL, 2018) e/ou outras estratégias heurísticas (XU; WANDELT; SUN, 2021).

Algoritmos Genéticos, pertencentes à classe de meta-heurísticas evolucionistas,

são sensíveis à maneira como é modelada a representação do problema, particularmente quanto à representação da solução candidata, especialmente em problemas combinatórios complexos (TALBI, 2009). Visando obter uma representação adequada para a utilização de um Algoritmo Genético que cumpra com as restrições do problema e obtenha soluções viáveis em tempo hábil, o algoritmo desenvolvido tem como propósito geral providenciar uma solução robusta para o problema integrado. Para além disso, tem como objetivo específico analisar o possível ganho de desempenho em termos de tempo de obtenção de uma solução ao paralelizar em GPU, utilizando a API CUDA, a implementação do algoritmo utilizado para a solução do modelo.

Foram realizados dois experimentos, com orçamentos computacionais diferentes, em sete cenários de simulação da operação semanal de uma empresa aérea fictícia. Comparado à literatura, para os três primeiros cenários simulados, o algoritmo proposto mostra uma melhora na velocidade de obtenção de uma solução em ambos os experimentos e consistência dos resultados em cada um dos cenários.

O foco deste trabalho é explorar o potencial de AGs paralelizados em CUDA como alternativa para solucionar o problema proposto, mostrando consistência nos resultados e o ganho de desempenho. Especificamente, o estudo busca encontrar uma representação adequada para a modelagem do problema, visando incorporar restrições do problema na representação.

No Capítulo 2 é discutida a literatura a respeito do *Airline Scheduling*, expondo trabalhos que lidam com os subproblemas que compõem o planejamento aéreo. Posteriormente, é apresentado o algoritmo genético clássico e variações de implementações paralelizadas, bem como o uso da API CUDA na paralelização do algoritmo. Por fim, é apresentado o modelo utilizado neste trabalho.

No Capítulo 3 é exposta a implementação do algoritmo proposto para a resolução do problema integrado de *Frequency Assignment Problem* e *Fleet Assignment Problem*, detalhando a representação adotada para a solução candidata do algoritmo genético, os operadores utilizados na busca do algoritmo e as estratégias para a implementação paralela em CUDA. No Capítulo 4 é feita a descrição dos experimentos computacionais realizados, apresentando a instância de dados utilizada e os parâmetros para o algoritmo genético

proposto. Os resultados são discutidos no Capítulo 4, Seção 4.5, onde uma discussão é conduzida a respeito dos resultados obtidos para este trabalho. Por fim, no Capítulo 5 é deliberado sobre os objetivos atingidos por este trabalho, destacando possíveis melhorias para o algoritmo proposto e trabalhos futuros.

2 Fundamentação Teórica

Neste Capítulo é discutido a respeito da literatura em relação a *Airline Scheduling*, com uma revisão de trabalhos que lidam com os subproblemas pertencentes à área. Posteriormente, é apresentado o Algoritmo Genético (AG) em sua forma clássica, além de variações para a implementação em paralelo do algoritmo. Na seção 2.2 é apresentada a modelagem do problema utilizada para o desenvolvimento deste trabalho.

2.1 Revisão da Literatura

O problema de alocação de frota é, de maneira geral, considerado como uma das "primeiras etapas"para o planejamento de um voo, com o roteamento de aviões sendo o "terceiro estágio", como destacado por Kenan, Jebali e Diabat (2018). Portanto, uma vez criada a programação do voo e definida uma frota para atendê-lo, é determinada a sequência de rotas e voos realizados por cada avião individualmente (BARNHART et al., 1998).

A incorporação da possibilidade de falhas e outras anomalias que podem ocorrer durante todo o processo, seja este o roteamento ou a alocação de frota, caracteriza "robustez"na modelagem. Robustez se refere à capacidade de recuperação de um plano perdido, como pode ser identificado nos trabalhos de He et al. (2023), Xu, Wandelt e Sun (2021) e Ben Ahmed et al. (2022), cada um implementando em seu modelo formas de assegurar a robustez da solução gerada.

Parte da definição da tripulação, modelada através dos problemas de Crew Rostering Problem (CRP) e Crew Pairing Problem (CPP), é explorada parcialmente pelo trabalho de Zeren, Özcan e Deveci (2024), que lida com o CPP.

O CPP consiste na definição de pareamentos de voos que começam e terminam com a mesma tripulação, cobrindo todos os voos de uma programação. O objetivo é minimizar os custos operacionais, como salário da tripulação, pernoites e "deadheads"(voos onde a tripulação são os passageiros para reposicionamento).

O CRP é a segunda etapa da otimização em relação à tripulação, utilizando os

voos criados pelo CPP para de fato atribuir os tripulantes aos voos. Tradicionalmente, ambos os problemas são resolvidos sequencialmente, primeiramente solucionando o CPP e em seguida o CRP. Contudo, Ouyang e Zhu (2023) explora a solução destes problemas de maneira integrada, incorporando ambas as etapas em um só problema de escalonamento, o *Crew Scheduling Problem*(CSP).

A Tabela 2.1 representa uma relação de trabalhos da literatura quanto ao(s) problema(s) solucionado(s), algoritmo utilizado e qual categoria pertencente do *Airline Scheduling*.

Tabela 2.1: Revisão da Literatura
ARP: Aircraft Routing Problem; **CPP:** Crew Pairing Problem; **CRP:** Crew Rostering Problem; **FAP:** Fleet Assignment Problem; **FAS:** Frequency Assignment Problem; **SD:** Scheduling Problem; **MIP:** Mixed Integer Linear Problem; **MINP:** Mixed Integer Non-Linear Problem; **CG:** Column Generation; **GA:** Genetic Algorithm; **MSGA:** Master-Slave Genetic Algorithm; **EA:** Evolutionary Algorithm; **CBC:** Coin or Branch-and-Cut Algorithm; **VNS:** Variable Neighbourhood Search ; **PS:** Proximity Search; **MA:** Memetic Algorithm; **AMPL:** A Mathematical Programming Language.

Category	**Literature**	**Problem(s)**	**Algorithm(s)**
Integrated Scheduling	Jesus, Nagano e Celestino (2023)	FAP + FAS	CBC
	Xu, Wandelt e Sun (2021)	FAP + ARP + SD + ROBUST	CG + VNS
	Ben Ahmed et al. (2022)	ROBUST + ARP + FAP + CPP	PS + C. SOLVER + AMPL
	Kenan, Jebali e Diabat (2018)	FAP + ARP + SD	CG
Crew Assignment	Ouyang e Zhu (2023)	CPP + CRP	GA
	Deveci e Demirel (2018)	CPP	EA + GA + MA
Fleet Assignment	Khanmirza, Nazarahari e Haghbeigi (2020)	SD + FAP	MSGA
Aircraft Routing	He et al. (2023)	ARP + ROBUST	CG

Jesus, Nagano e Celestino (2023) lida com os problemas de Alocação de Frota (FAP) e Determinação da frequência de voos (FAS) de maneira integrada. O modelo matemático proposto busca maximizar o lucro da empresa aérea, considerando uma janela de tempo de múltiplos turnos de trabalho, em um formato de planejamento ou semanal ou diário, aplicando esse modelo em uma companhia aérea fictícia que utiliza o Aeroporto Internacional de Goiânia como sede de operações, selecionando as rotas que irão compor a rede e determinando o tamanho ótimo da frota para operar.

Xu, Wandelt e Sun (2021) propõe uma solução para o Airline Scheduling, composto pelos problemas de roteamento de aviões (ARP) e alocação de frota (FAP) integrados, com ênfase em robustez. Neste contexto, robustez é caracterizada pela consideração de possíveis disrupções no planejamento, como atrasos propagados, focando em estabilidade e recuperabilidade dos planos gerados, combinando geração de coluna e busca local variada para solucionar o modelo.

Ben Ahmed et al. (2022) visa múltiplas facetas do problema integrado de Airline Scheduling, buscando solucionar tanto os problemas de ARP e FAP quanto o pareamento de tripulação (CPP). Considera robustez no planejamento ao priorizar planos onde uma equipe permanece num mesmo avião, propondo um modelo não linear de arcos para as rotas e o pareamento de tripulação. As rotas e o pareamento da tripulação possuem um grafo separado, sendo a função objetivo maximizar o lucro arrecadado menos o custo da utilização da frota, decompondo o modelo não linear em pequenos modelos lineares usando técnicas de linearização e os solucionando de maneira "relaxada".

Kenan, Jebali e Diabat (2018) trata do Flight Scheduling, nome dado aos problemas integrados de FAP e ARP, propondo um modelo que leva em consideração voos opcionais e atrasos. O objetivo é maximizar o lucro determinado pela diferença entre o montante esperado e a soma dos diferentes gastos envolvidos, incluindo os gerados por atrasos. Contudo, devido à complexidade do modelo proposto, não foi possível solucioná-lo de maneira determinística, sendo necessária uma reformulação do problema em um modelo de colunas que é solucionado por meio de Geração de Colunas.

Ouyang e Zhu (2023) lida com o Crew Scheduling Problem (CSP), dado pela junção dos problemas de CPP e CRP, que geralmente são solucionados de maneira sequencial. Os autores propõem uma solução integrada para os problemas, utilizando metaheurísticas, particularmente um Algoritmo Genético (AG) paralelizado. O modelo proposto considera apenas pilotos, isto é, a tripulação do *cockpit*, argumentando que o restante da tripulação possui diferentes restrições para serem modeladas.

Deveci e Demirel (2018), por sua vez, trata apenas do problema de CRP, focando em demonstrar a aplicação de técnicas de Computação Evolucionista para a solução do problema. No trabalho é implementado um AG padrão, um AG modificado que aplica heurísticas locais para melhorar a solução inicial e na nova geração de indivíduos gerada a cada iteração e, por fim, um Algoritmo Memético (*Memetic Algorithm*) que mantém as melhorias do AG modificado e adiciona mais heurísticas de busca para aprimorar a solução inicial e as subsequentes.

Khanmirza, Nazarahari e Haghbeigi (2020) lida apenas com os problemas de *scheduling design* e FAP, propondo um modelo que realiza a alocação de frota para o

melhor itinerário e fatora a recaptura de demanda, capturando mais passageiros em locais onde a demanda é mais alta. Optaram por utilizar um PMS-GA (*Parallel master–slave Genetic Algorithm*), uma modificação de um algoritmo genético em paralelo, distribuindo entre os módulos *master* e *slave* as decisões necessárias para o modelo proposto.

A literatura exibe um repertório de abordagens diferentes para os problemas relacionados ao *Airline Scheduling*, com a programação linear e não-linear de inteiros mista sendo a principal forma de modelagem dos subproblemas, como o Roteamento de Aviões e o Escalonamento de Tripulação. Embora resolvedores comerciais sejam frequentemente empregados para estas formulações, estratégias de meta-heurísticas, incluindo Algoritmos Genéticos, que possuem destaque na solução do problema de escalonamento de tripulação (Ouyang e Zhu (2023), Deveci e Demirel (2018)) ou na integração de problemas como o *scheduling design* e FAP (Khanmirza, Nazarahari e Haghbeigi (2020)). Quanto ao subproblema integrado da Alocação de Frota e Determinação de Frequência de Voos, a literatura acerca do uso de AGs e outras estratégias evolutivas para solucioná-lo é restrita.

2.1.1 Algoritmo Genético

Um algoritmo genético pertence a uma gama de estratégias meta-heurísticas bioinspiradas da chamada computação evolucionista, que toma como princípios ideias de Charles Darwin (GOLDBERG, 1989) a respeito da evolução de espécies, baseado na sobrevivência do mais apto.

De maneira geral, algoritmos evolucionários são iterativos, com cada iteração criando uma nova geração de indivíduos, cada um representando uma possível solução dentro de uma população de soluções. A cada iteração, é realizada a aplicação de operadores para explorar o espaço de busca de soluções ou aperfeiçoar soluções já existentes, sendo esses operadores de seleção, crossover e mutação:

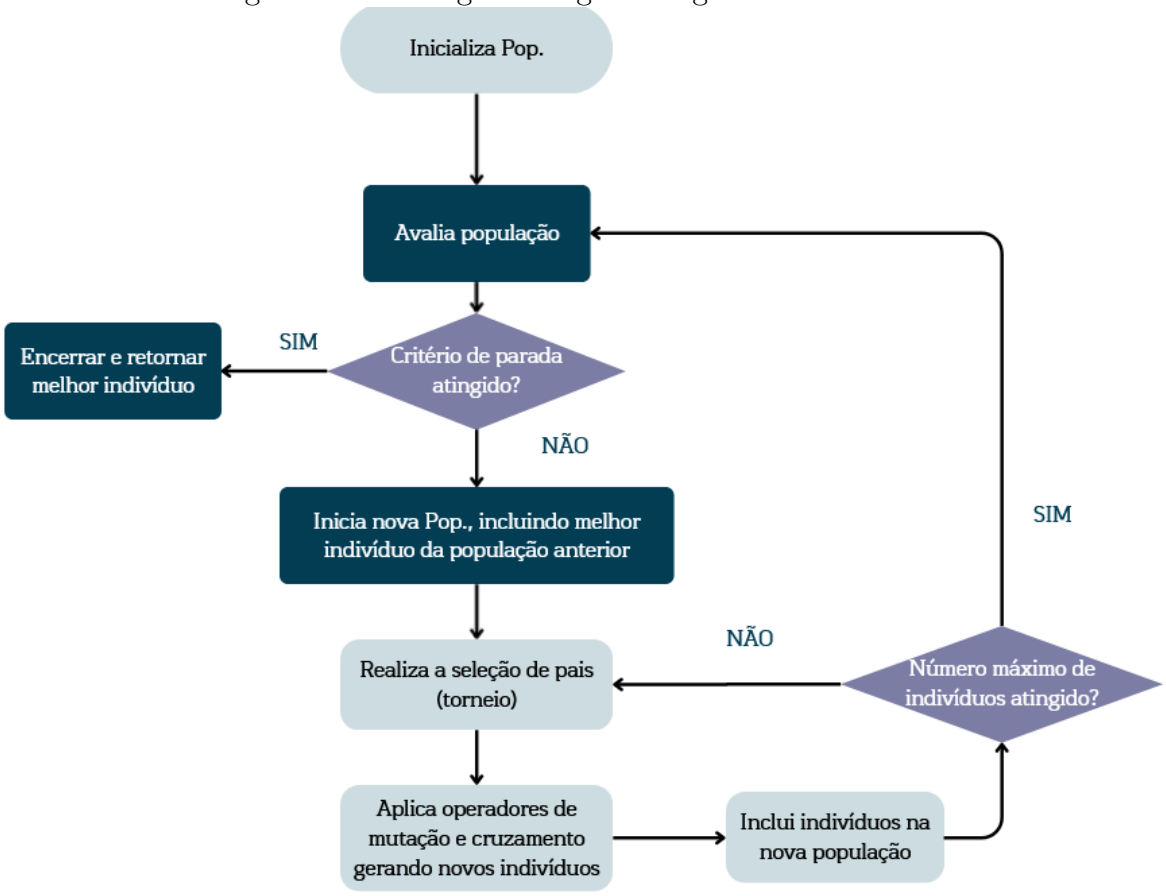
- **Seleção**: A cada iteração, este operador seleciona indivíduos pais da população para a aplicação dos operadores de crossover e mutação. Dentre as possíveis implementações, pode-se citar a escolha aleatória de dois indivíduos ou a seleção torneio.
- **Crossover**: Havendo dois indivíduos pais, o operador de crossover gera um novo indíviduo através da recombinação da solução dos indivíduos pais. Como exemplo,

o crossover *single-point* seleciona um ponto da solução dos indivíduos pais e realiza uma troca entre os cromossomos.

- **Mutação**: Este operador aplica uma pertubação na solução de um indivíduo, podendo ser uma pertubação exploratória (inclui maior diversidade de soluções) ou para aperfeiçoar soluções já existentes.

A Figura 2.1.1 representa o fluxograma do funcionamento do AG clássico.

Figura 2.1: Fluxograma algoritmo genético clássico.



O algoritmo 1 representa o funcionamento do Algoritmo Genético Clássico.

Dentre as diversas possíveis modificações ao algoritmo original, a prática de elitismo na população, como descrito por Petrowski e Ben-Hamida (2017) é uma forma de manter o melhor (ou os melhores) indivíduos da população anterior na nova população.

Algoritmo 1: Algoritmo Genético Clássico.

Saída: Melhor indivíduo

início
  geracao ← 0
  Pop ← Gera população inicial
  Aval(Pop) Avalia população inicial
  Ordena(Pop)
  Melhor ← Pop[0]
  repita
    geracao ← geracao + 1
    NovaPop ← ϵ
    repita
      $P_1 \leftarrow Selecao(Pop)$
      $P_2 \leftarrow Selecao(Pop)$
      Realiza crossover entre P_1 e P_2 para gerar filhos B_1 e B_2
      Aplica mutações em B_1 e B_2 para gerar C_1 e C_2
      NovaPop ← $NovaPop \bigcup \{C_1\}$
      NovaPop ← $NovaPop \bigcup \{C_2\}$
    até $TamNovaPop = TamPop$;
    Pop ← $NovaPop$
    Aval(Pop)
    Ordena(Pop)
    se $Melhor < Pop[0]$ **então**
      Melhor ← Pop[0]
    fim se
  até $geracao = MaxIteracoes$;
  retorna *Melhor*
fim

2.1.2 Algoritmo Genético Paralelo

De maneira geral, algoritmos genéticos tendem a ser implementados para solucionar problemas da classe NP-Difícil, não plausíveis de serem solucionados usando métodos convencionais. Problemas onde o conhecimento *a priori* não existe também encontram uso favorável de algoritmos genéticos, proporcionando soluções rápidas (GHAZFAN; BALA; NOLAN, 1994).

Considerando a natureza da busca realizada por um algoritmo genético, que mantém diversas soluções em paralelo, AGs são candidatos a serem paralelizados em sua implementação (ARORA; TULSHYAN; DEB, 2010a), ganhando um aumento em velocidade enquanto retêm as mesmas características de uma busca realizada por um AG em sequencial. Os chamados MSGA (*Master-Slave Genetic Algorithm*) são implementações de AGs utilizando computação paralela, podendo ser divididos em quatro tipos (CANTú-

PAZ, 2001):

- **Síncrono:** A avaliação dos indivíduos é realizada em paralelo, de maneira distribuída, enquanto o núcleo principal aplica os operadores genéticos e gera a nova população. É necessário que todos os indivíduos tenham finalizado a avaliação para a geração da nova população, sendo este o passo de sincronia.

- **Assíncrono:** Similar ao síncrono, porém permite que indivíduos que já terminaram sua avaliação retorne ao fluxo principal para que novos indivíduos possam ser gerados e posteriormente avaliados, não necessitando do passo de sincronia.

- **Distribuído:** Neste, a população está localizada em um espaço compartilhado de memória, onde cada núcleo aplica os operadores genéticos e avalia os indivíduos de maneira assíncrona.

- **Rede:** Um número de AGs são executadas de maneira independente, com memórias, núcleos, operadores genéticos e avaliação independentes, sendo a população composta por subgrupos localizados em cada AG. O melhor individuo de cada geração de uma AG independente é enviado para as outras através de uma rede compartilhada.

Dentre os quatro tipos, o MSGA síncrono é o de menor complexidade para implementação, sendo necessário apenas um passo de sincronia para o término da avaliação da população em paralelo.

A aceleração de Algoritmos Genéticos por meio de paralelização em GPUs com CUDA é uma abordagem explorada na literatura. O trabalho de Arora, Tulshyan e Deb (2010b) oferece uma contribuição ao realizar uma análise aprofundada dos gargalos de desempenho e das otimizações necessárias nos operadores genéticos. O estudo demonstra como o ajuste fino dessas operações explora o potencial da arquitetura paralela e obtém ganhos de velocidade significativos.

A seguir, será apresentada a modelagem para o problema, descrevendo os conjuntos utilizados, a definição formal do modelo e a descrição informal do mesmo.

2.2 Modelo

Nesta seção, será apresentada a modelagem utilizada para o problema integrado de FAP e FAS, com uma breve descrição do problema, seguida da definição formal do modelo. Por fim, é realizada uma descrição informal do problema.

2.2.1 Descrição do Problema

O problema de Alocação de Frota busca designar quais tipos de aeronaves devem compor a frota de cada voo do cronograma de uma companhia aérea, visando maximizar o lucro operacional, enquanto o problema de Determinação de Frequência define quantas vezes os voos de uma determinada rota devem ser operados dentro do período de planejamento para maximizar os lucros.

O modelo utilizado integra ambos os problemas, portanto, definindo quais voos incluir no cronograma, a frota a ser utilizada e com que frequência operá-los. Para este trabalho será utilizado o modelo proposto por Jesus, Nagano e Celestino (2023):

A seguir, uma descrição dos conjuntos, variáveis e parâmetros utilizados no modelo.

Conjuntos:

$L \leftarrow$ conjunto de todas as rotas consideradas, conectando o hub aos destinos, $l \in L$.

$W \leftarrow$ conjunto de todas as janelas de tempo / turnos considerados no planejamento semanal, $w \in W$.

$A \leftarrow$ conjunto de todos os tipos de aeronaves considerados no modelo, $a \in A$.

$E \leftarrow$ conjunto de arestas orientadas, subconjunto de L, para restrições de transbordo.

$IN \leftarrow$ subconjunto de E, indicando entrada de nó da janela de tempo anterior.

$OUT \leftarrow$ subconjunto de E, indicando saída de nó na janela de tempo presente.

Variáveis:

$Bin_{1,a} \leftarrow$ variável binária para aplicações de decisão de contingência relacionadas ao número total de tipos de aeronaves considerados no modelo, $\forall a \in A$.

$Bin2_{l,a} \leftarrow$ variável binária para aplicações de decisão de contingência, restringindo destinações que a aeronave do tipo a não possui alcance suficiente, $a \in A, l \in L$.

$F_{l,w,a} \leftarrow$ frequência de voos para o destino l na janela de tempo w para a aeronave do tipo a, $a \in A, w \in W, l \in L$.

$N_a \leftarrow$ número / tamanho da frota necessária para a aeronave do tipo $a, a \in A$

$P_{l,w,a} \leftarrow$ número de passageiros transportados para a destinação l durante a janela de tempo w pela aeronave do tipo a, $a \in A, w \in W, l \in L$.

Parâmetros:

$C_{l,a} \leftarrow$ CASK para o destino l operado pela aeronave do tipo a.

$D_l \leftarrow$ distância até o destino de voo l.

$K1, K2 \leftarrow$ constantes que representam o número máximo de tipos distintos de aeronaves na rede e em cada rota, respectivamente.

Parâmetros Adicionais:

$M \leftarrow$ parâmetro Big-M, que representa um limite superior para a frequência de voos.

$Q_{l,w} \leftarrow$ demanda de passageiros para o destino l durante a janela de tempo w.

$R_a \leftarrow$ alcance máximo da aeronave do tipo a.

$S_a \leftarrow$ número de assentos disponíveis na aeronave do tipo a.

$T_{l,a} \leftarrow$ preço da passagem para passageiros viajando para o destino l com a aeronave do tipo a.

$$\max \sum_{i \in I} \sum_{w \in W} \sum_{a \in A} T_{i,a} P_{i,w,a} - C_{i,a} D_i S_a F_{i,w,a} \tag{2.1}$$

$$\sum_{a \in A} P_{i,w,a} \leq Q_{i,w}, \quad \forall i \in L, w \in W; \tag{2.2}$$

$$P_{i,w,a} \leq S_a F_{i,w,a}, \quad \forall i \in L, w \in W, a \in A; \tag{2.3}$$

$$\sum_{i \in OUT} F_{i,w,a} = \sum_{i \in IN} F_{i,w-1,a}, \quad \forall OUT \in E, IN \in E, w \in W, a \in A; \tag{2.4}$$

$$\sum_{i} F_{i,w,a} \leq N_a, \quad \forall w \in W, a \in A; \tag{2.5}$$

$$Bin_{1,a} \leq K1; \tag{2.6}$$

$$F_{i,w,a} \leq M Bin_{1,a}, \quad \forall i \in L, w \in W, a \in A; \tag{2.7}$$

$$\sum_{a \in A} Bin_{2,i,a} \leq K2, \quad \forall i \in L; \tag{2.8}$$

$$F_{i,w,a} \leq M Bin_{2,i,a}, \quad \forall i \in L, w \in W, a \in A; \tag{2.9}$$

$$R_i \geq D_i Bin_{2,i,a}, \quad \forall i \in L, a \in A; \tag{2.10}$$

$$F, P, N \in \mathbb{Z}; \tag{2.11}$$

$$Bin_{1,a}, Bin_{2,i,a} \in \{0, 1\}. \tag{2.12}$$

2.2.2 Descrição do Modelo

A função objetivo 2.1 visa maximizar o lucro obtido, considerando sua programação de voos. Para tanto, é subtraído do ganho de receita obtido dos passageiros transportados o custo operacional em cada rota. O custo operacional engloba a distância percorrida até o destino, a frequência de voos, a capacidade máxima do modelo de avião operado e o CASK (*cost of available seat kilometer*). CASK representa o custo de cada assento por quilômetro para um determinado destino, associado a um modelo de aeronave. A restrição 2.2 limita o somatório do número de passageiros transportados em uma dada rota, turno e tipo de aeronave à demanda da rota no turno. A restrição 2.3 dita que o número de assentos ofertados em cada rota deve ser maior ou igual ao número de passageiros transportados. A restrição 2.4 indica que o fluxo de aeronaves partindo de um aeroporto deve ser igual ao fluxo de aeronaves chegando no mesmo aeroporto em um turno anterior (janela

de tempo anterior). Neste modelo, uma aeronave pode permanecer no aeroporto caso seja uma decisão lucrativa. A restrição 2.5 indica que o número total de aeronaves necessárias para suprir a demanda de voos pode ser determinado. As restrições de diversidade de rede 2.6 e 2.7 permitem a introdução de um limite estratégico para o número total de tipos de aeronaves diferentes operando sobre a rede de uma companhia. As restrições 2.8 e 2.9 permitem limitar o número total de tipos de aeronaves em rotas específicas. A restrição 2.10 permite que um tipo de aeronave opere sobre uma rota somente se possuir alcance suficiente para cobrir a distância do voo. O objetivo é maximizar o lucro operacional da companhia aérea, alocando a frota necessária, a quantidade de passageiros transportada e a frequência da realização de voos para cada dia, considerando um planejamento semanal com 4 turnos diários.

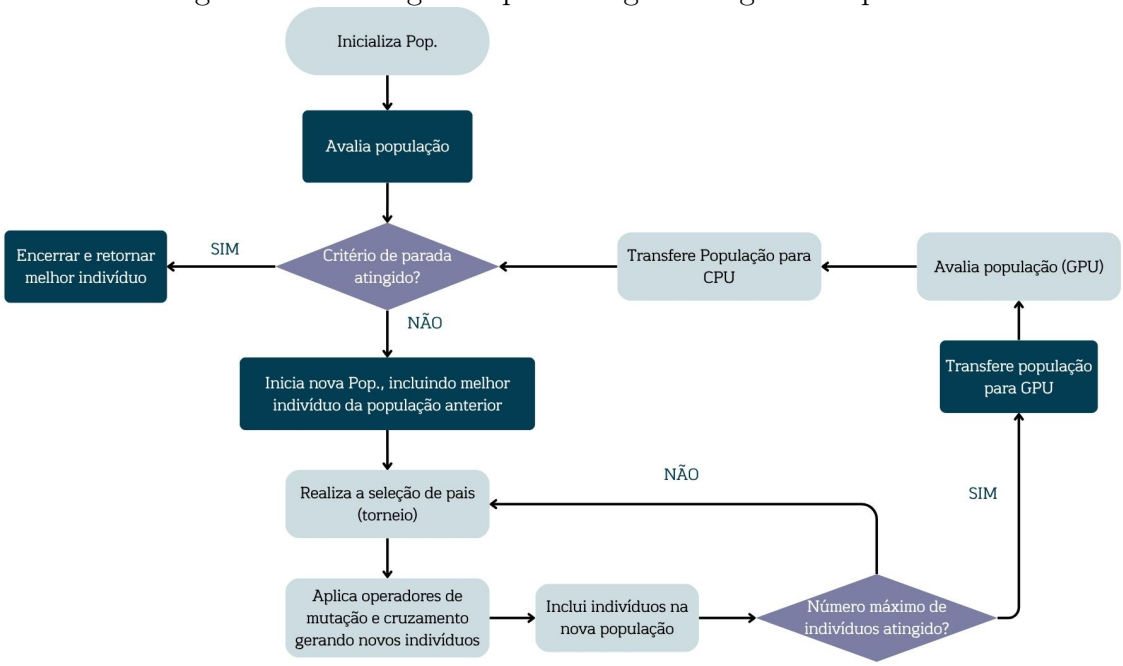
3 Algoritmo Genético Proposto

Neste Capítulo, é apresentado o algoritmo genético proposto para este trabalho, inicialmente representando o fluxograma para o funcionamento do algoritmo, em seguida são descritos detalhes técnicos em relação a representação da solução e dos operadores genéticos implementados, assim como adaptações realizadas para a implementação em CUDA.

3.1 Fluxograma

A Figura 3.1 mostra o funcionamento do algoritmo, partindo da geração inicial e à geração de novas soluções candidatas ao longo das iterações.

Figura 3.1: Fluxograma para o algoritmo genético paralelo.

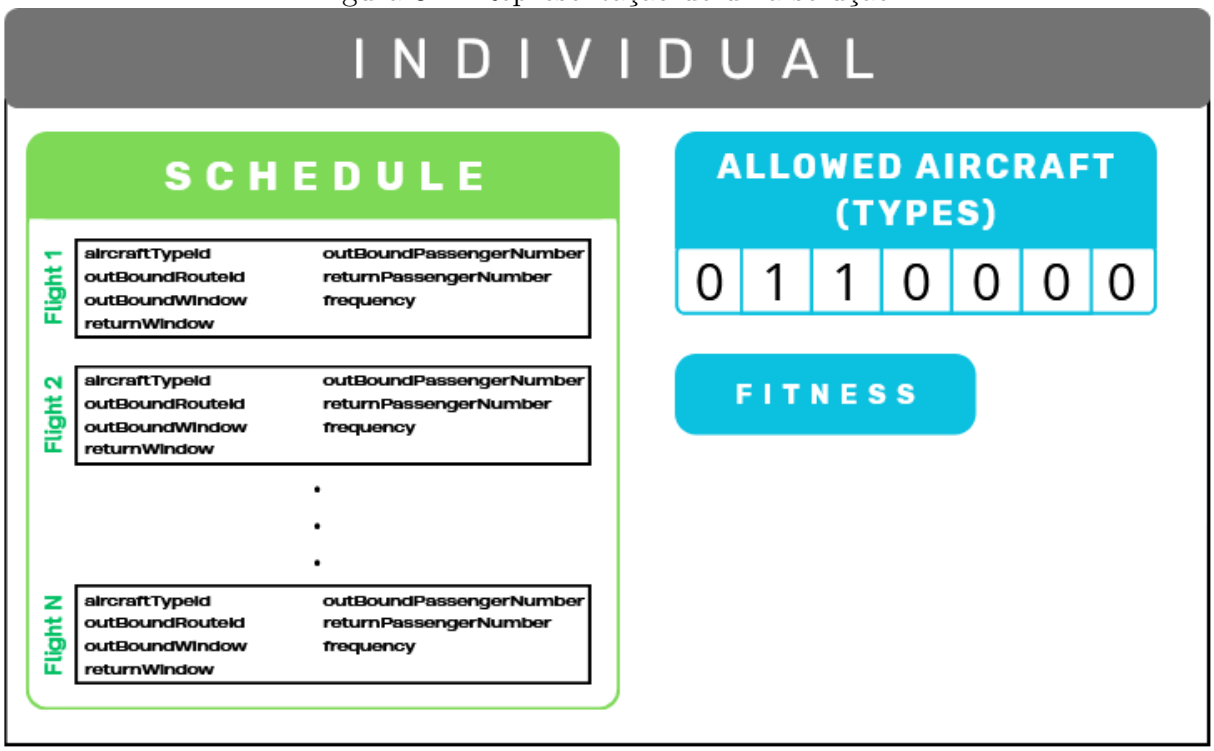


3.2 Implementação do Algoritmo

Para este trabalho, a implementação do algoritmo genético para solucionar o problema integrado foi feita na linguagem de programação C++, paralelizado em GPU com o uso

da biblioteca CUDA (NVIDIA; VINGELMANN; FITZEK, 2020). Neste trabalho, foi realizada a implementação de uma MSGA síncrona (2.1.2). Cada cromossomo de um indivíduo, isto é, uma solução em potencial da população, é representado como ilustrado na Figura 3.2.

Figura 3.2: Representação de uma solução.



Na representação ilustrada na Figura 3.2, *Schedule* é um vetor composto por uma estrutura de dados *Flight*. A estrutura de dados contém informações acerca de um voo realizado, possuindo a frequência dos voos (*frequency*), o modelo de avião utilizado (*aircraftTypeId*), a rota cumprida (*outBoundRouteId*), a janela de tempo de ida (*outBoundWindow*), a janela de tempo para o voo de volta (*returnWindow*), a quantidade de passageiros no voo de ida (*outBoundPassengerNumber*) e a quantidade de passageiros no voo de volta (*returnPassengerNumber*). Esta representação proporciona a vantagem de satisfazer a restrição 2.5, uma vez que reúne a frequência do voo de ida e volta numa mesma estrutura, indicando apenas as janelas de ida e volta. *Fitness* indica a aptidão do indivíduo, isto é, o lucro obtido a partir do cálculo da função objetivo como descrito no modelo (2.2). *Allowed Aircraft* é um vetor binário que indica quais modelos de aviões estão disponíveis para uso na composição de uma frota. A representação de um

voo dada pela estrutura de dados *Flight* lida simultaneamente o voo de ida e o voo de volta, utilizando-se da informação do turno de ida e volta para um par origem-destino. O modelo 2.2 considera a topologia *hub-to-spoke*, isto é, os voos possuem conexão direta apenas entre o aeroporto *HUB* e os *spokes* (COOK; GOODWIN, 2008).

3.2.1 Operadores

Para o *crossover*, foi realizada uma troca simples entre os modelos disponíveis de frota, sorteando um índice que, partindo do índice escolhido de modelos disponíveis, haverá a troca entre os genes dos pais, sendo este um crossover de um ponto.

O algoritmo não implementa o intercâmbio de rotas pois, via experimentação, determinamos que, em média, os resultados são melhores quando não há a mesclagem de programações entre os indivíduos pais, devido à disponibilidade de frota. Um indivíduo, no pior caso, poderia receber uma programação onde todos os voos estão sendo realizados por um modelo de avião no qual não está disponível para compor a frota de sua programação. Do ponto de vista operacional, isto pode gerar uma solução inválida, pois não possui nenhum voo programado ou até mesmo receber uma programação conflitante, com dois ou mais voos cumprindo a mesma janela de tempo, porém violando a restrição de demanda (2.3).

Para a mutação, o algoritmo possui cinco operadores diferentes:

- ***Flip Aircraft Type***: Realiza um "flip"no bit referente a disponibilidade de um modelo de avião.

- ***Add Route***: Sorteia, entre as viagens disponíveis a serem realizadas, um turno de ida e um turno de volta, alocando passageiros para a ida e volta de acordo com a demanda disponível, além do modelo de aeronave operante. Essa alocação considera a demanda já cumprida no turno. Para determinar a quantidade de passageiros para a ida e a volta, possuímos duas variações de alocação de acordo com a quantidade disponível de modelos de aviões:

  - Caso somente um modelo esteja disponível: O novo voo inserido irá cumprir pelo menos 20% da demanda de ambos turno de ida e volta

  - Caso mais de um modelo está disponível: O novo voo inserido tentará atender completamente a demanda do turno. Caso a demanda seja superior a capacidade do avião selecionado, será alocado a quantidade de passageiros de acordo com o limite de capacidade.

- ***Remove Route***: Seleciona aleatoriamente uma viagem presente na programação de um indivíduo e a remove.

- ***Alter Frequency***: Seleciona aleatoriamente uma viagem presente na programação de um indivíduo e tenta alterar a frequência de voos, seja aumentando a frequência ou diminuindo.

- ***Adjust Passengers***: Seleciona um voo presente na programação do indivíduo e ajusta a quantidade de passageiros tanto no voo de ida quanto no de volta, aumentando a quantidade de passageiros de acordo com a demanda disponível em cada janela de tempo considerando a demanda já atendida por todos os modelos que estão operando sobre o voo selecionado.

Uma vez selecionado, o indivíduo pode receber um dos cinco operadores de mutação descritos, sendo esta escolha estocástica baseada no sorteio de um número entre 0 e 4.

Após um crossover/mutação, é realizado um reparo nas novas soluções geradas para manter a viabilidade da solução quanto às restrições de demanda 2.3 e frota permitida 2.6 2.7, removendo possíveis voos inválidos ou voos vazios.

Para a seleção dos indivíduos para aplicação dos operadores, implementamos um torneio (TALBI, 2009) de tamanho 3 (tamanho definido de forma experimental), onde são sorteados três indivíduos da população e selecionados os dois melhores para gerar a próxima geração.

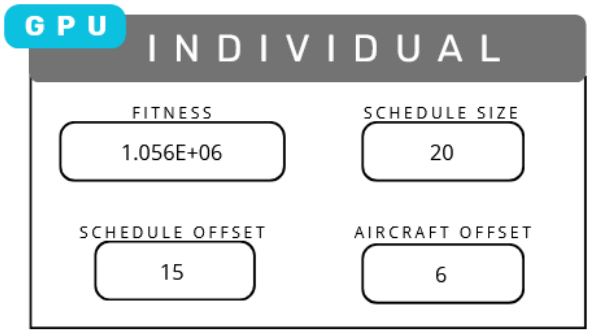
Os indivíduos da população, inicialmente, não possuem rotas disponíveis nem informação sobre quais modelos podem compor sua frota. Para preencher este indivíduo, é realizada a aplicação do operador de mutação N vezes sobre o indivíduo, sendo N um número sorteado aleatoriamente. A mutação aplicada é sorteada entre os cinco operadores descritos anteriormente (3.2.1), com o *Flip Aircraft Type* e o *Add Route* sendo os principais operadores para inicializar a construção da solução de um indivíduo. Não foi implemen-

tado um algoritmo construtivo guloso para as soluções iniciais pois, experimentalmente, comparando com a inicialização através de mutações, a população inicial foi menos diversa em termos de soluções candidatas, criando uma tendência na busca, desacelerando e convergindo prematuramente em mínimos locais distantes das melhores soluções encontradas pela implementação com mutações. A nova população para cada geração é feita pela troca parcial da população original, havendo um elitismo simples, onde o melhor indivíduo em termos de aptidão da população anterior é inserido na nova população, mantendo assim uma qualidade crescente da solução em cada iteração do algoritmo.

3.2.2 Implementação paralela

Para a realização da implementação do MSGA (2.1.2), algumas estruturas de dados utilizadas na CPU foram adaptadas, principalmente a estrutura *Individual* que, para a GPU, é representada na Figura 3.3.

Figura 3.3: Representação de um indivíduo na GPU.



O indivíduo é simplificado, não possuindo mais os vetores *Schedule* e *AllowedAircraft*. É realizada a adição de dois índices, *schedule offset* e *aircraft offset* que apontam para duas estruturas de dados centralizadoras, alocadas na GPU. Todas as programações de todos os indivíduos estão alocadas em um único vetor *AllSchedules* na GPU, representado pela Figura 3.4, assim como as informações a respeito de quais modelos de aviões estão disponíveis em cada indivíduo estão presentes em um único vetor *AllAllowedAircraft*, representado na Figura 3.5. Portanto, para acessar os dados de programação pertencentes a um indivíduo $x, x \in POP$ com POP sendo a população, basta utilizar o índice *schedule offset* presente no indivíduo e acessar sua programação através do vetor *AllSchedules*, de maneira análoga para o índice *aircraft offset* e o vetor *AllAllowedAircraft*.

A estruturação foi feita desta forma para simplificar a transferência da população para a GPU, evitando a necessidade de alocar memória de um vetor de estruturas de dados e outras conversões para garantir que não há memória sendo utilizada desnecessariamente ou alocada incorretamente. Quanto à alocação dos dados sobre a instância, uma vez definido que haverá a execução do módulo em paralelo, os dados são alocados uma única vez e mantidos em memória na GPU, evitando a necessidade de repetidas alocações de dados estáticos.

Figura 3.4: Agrupamento de todas as programações na GPU.

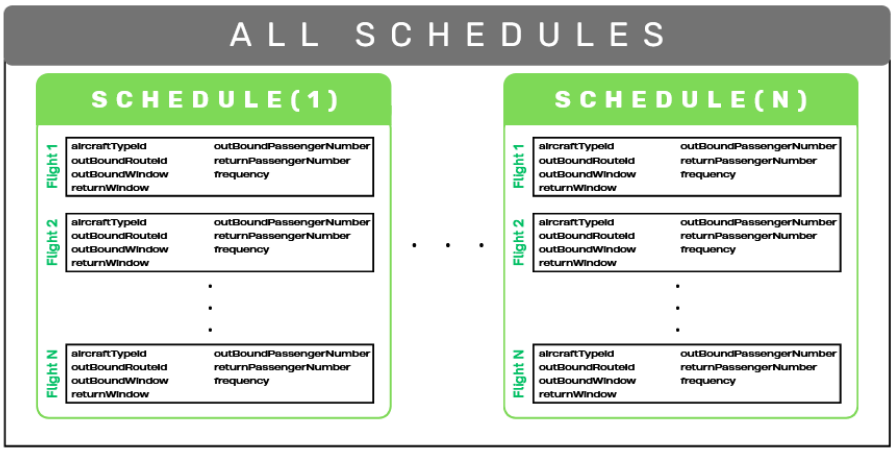
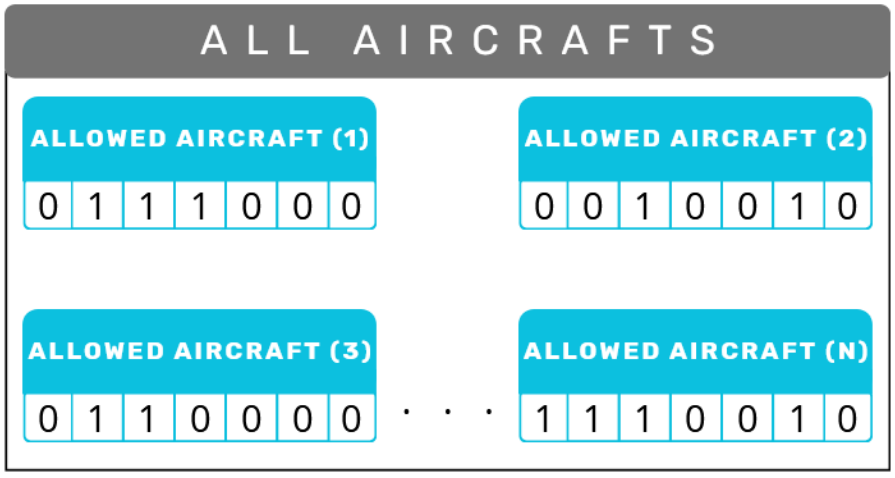


Figura 3.5: Agrupamento de todos os modelos de aviões disponíveis para cada indivíduo na GPU.



4 Experimentos Computacionais

Para os experimentos, é feita uma comparação entre os resultados obtidos pelos autores do artigo (JESUS; NAGANO; CELESTINO, 2023) dado os cenários de simulação operacional. No artigo, são descritos 7 cenários diferentes onde o número de modelos de aviões disponíveis para a composição da frota é limitado:

- Cenário 1 - Apenas um modelo de avião;
- Cenário 2 - Até dois modelos de aviões;
- Cenário 3 - Até três modelos de aviões.
- Cenário 4 - Até quatro modelos de aviões.
- Cenário 5 - Até cinco modelos de aviões.
- Cenário 6 - Até seis modelos de aviões.
- Cenário 7 - Até sete modelos de aviões.

Para cada um dos cenários, realizamos dois experimentos:

- Cinco execuções independentes do AG, com 5000 iterações, tanto em sequencial quanto em paralelo, obtendo o melhor resultado das 5 execuções e a média de tempo para obtenção destes resultados utilizando uma seed fixa 10 para a geração de números aleatórios;
- Dez execuções independentes do AG, com 2000 iterações, repetindo o mesmo procedimento em paralelo e em sequencial utilizando uma seed fixa de 320;

O objetivo é demonstrar que, mesmo com um orçamento computacional diferente para cada experimento, como um orçamento menor para o conjunto 4 e um maior para o conjunto 4, os resultados obtidos permanecem consistentes, variando o tempo de execução.

Dos parâmetros para a busca, a população foi composta por 128 indivíduos. Em termos de operadores genéticos, foram utilizados os operadores de cruzamento e mutação,

com as respectivas taxas de cruzamento e mutação 80% e 40%, definidas de maneira empírica ao realizar os experimentos.

Utilizamos a biblioteca $< random >$ para geração de números, usando as funções de distribuição uniforme inteira e real, usando um *mersenne twister* 19937 como pseudo-gerador de números aleatórios.

4.1 CUDA

Para a avaliação dos indivíduos na estrutura CUDA, utilizamos 128 *threads* para realizarem a execução em paralelo por bloco, definido experimentalmente, enquanto o número de blocos é o quádruplo da quantidade de multi-processadores presentes na GPU utilizada, no caso, 24 multi-processadores, totalizando 96 blocos. O objetivo é manter o maior número de multi-processadores ocupados, a fim de maximizar o uso da GPU. A função objetivo 2.1 não recebeu modificações significativas, exceto no que se diz respeito ao acesso de estrutura de dados que antes estavam localizadas na estrutura do indivíduo e receberam um *flattening*, sendo agrupadas nos mega vetores mencionados anteriormente (3.4 3.5). Ou seja, enquanto na avaliação da aptidão dos indivíduos na CPU basta acessar diretamente os vetores que contêm a programação de voos e os modelos utilizados na frota, na GPU é necessário, primeiramente, resgatar, utilizando um índice, o início do gene do indivíduo no vetor de programação de voos e no vetor de frota permitida.

É importante ressaltar que as operações matemáticas realizadas em GPU podem divergir em termos de casas decimais das realizadas em sequencial na CPU, fazendo-se necessária a execução em sequencial e em paralelo mencionadas *a priori* (4 4) devido à sensibilidade do algoritmo de busca com a propagação dessa possível diferença.

4.2 Arquitetura Computacional

Os experimentos foram realizados em uma máquina com a configuração a seguir:

- Processador - i5-10400, 12MB cache, 2.9 Ghz (4.30 Ghz max)
- Memória principal - 16 GB (2x8 GB) DDR4 2666Mhz

- Placa de vídeo - RTX 4060 8GB VRAM
- Sistema Operacional - Windows 11

4.3 Código Fonte

O código fonte com a implementação está localizado em ⟨https://github.com/nicolassoam/TCC⟩. Os dados da instância podem ser solicitados aos autores do artigo Jesus, Nagano e Celestino (2023).

4.4 Instância

Utilizamos, para os experimentos, os dados para simulação coletados por Jesus, Nagano e Celestino (2023), com informações reunidas da ANAC (Agência Nacional de Aviação Civil) e de entrevistas com Figuras da área. Foram reunidos dados de 10 destinos de aeroportos distintos, formando, portanto, 20 viagens diferentes, considerando os pares de origem-destino entre o "HUB"e seus "spokes".

A Tabela 4.1 descreve os modelos de aeronaves quanto à capacidade máxima de passageiros permitida (JESUS; NAGANO; CELESTINO, 2023).

Tabela 4.1: Capacidade máxima de passageiros por modelo de aeronave.

Aeronave	**Capacidade**
ATR72–600	68
Embraer E190-E2	106
Airbus A220–110	115
Embraer E195-E2	132
Airbus A319neo	140
Airbus A220–300	141
Boeing 737 MAX 7	156

4.5 Resultados

Nesta seção, são apresentados os resultados obtidos a partir da execução dos experimentos descritos anteriormente (4 4).

As Tabelas 4.2 e 4.3 representam o número de aeronaves por modelo em cada um dos cenários descritos, com a Tabela 4.2 representando os resultados obtidos para o primeiro conjunto de experimentos e a Tabela 4.3 representando os resultados para o segundo conjunto.

Tabela 4.2: Número de aeronaves para cada cenário com 5 execuções independentes e 5000 iterações.

Aeronave	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
ATR72-600	0	0	1	1	1	1	1
E190-E2	0	8	7	7	7	7	7
A220-100	11	3	3	3	3	3	3
E195-E2	0	0	0	0	0	0	0
A319neo	0	0	0	0	0	0	0
A220-300	0	0	0	0	0	0	0
737 MAX 7	0	0	0	0	0	0	0
Total	11	11	11	11	11	11	11
Lucro (US$)	7.794.570	7.849.540	7.884.380	7.884.380	7.884.380	7.884.380	7.884.380

Tabela 4.3: Número de aeronaves para cada cenário com 10 execuções independentes e 2000 iterações.

Aeronave	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
ATR72-600	0	0	1	1	1	1	1
E190-E2	0	8	7	7	7	7	7
A220-100	11	3	3	3	3	3	3
E195-E2	0	0	0	0	0	0	0
A319neo	0	0	0	0	0	0	0
A220-300	0	0	0	0	0	0	0
737 MAX 7	0	0	0	0	0	0	0
Total	11	11	11	11	11	11	11
Lucro (US$)	7.794.570	7.849.540	7.884.380	7.884.380	7.884.380	7.884.380	7.884.380

As Tabelas 4.4 e 4.5 representam a atribuição de frequência de voos por destino para cada um dos cenários, com a Tabela 4.4 representando os resultados obtidos para o primeiro conjunto de experimentos e a Tabela 4.5 representando os resultados para o segundo conjunto.

As Tabelas 4.6 e 4.7 representam um comparativo do tempo de execução entre a CPU, GPU e a Literatura. A Tabela 4.6 representa os resultados para o primeiro conjunto de experimentos, enquanto a Tabela 4.7 representa os resultados para o segundo.

As Tabelas 4.8 e 4.9 representam um comparativo entre o lucro obtido por este trabalho e a literatura.

Tabela 4.4: Atribuição de frequência para cada cenário com 5 execuções independentes e 5000 iterações.

Destino	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
SBGR	22	21	21	21	21	21	21
SBSP	21	21	21	21	21	21	21
SBRJ	14	14	14	14	14	14	14
SBCF	14	14	14	14	14	14	14
SBKP	7	7	7	7	7	7	7
SBGL	7	7	7	7	7	7	7
SBPA	7	7	7	7	7	7	7
SBCY	7	7	7	7	7	7	7
SBPJ	7	7	7	7	7	7	7
SBCT	7	7	7	7	7	7	7
Total	113	112	112	112	112	112	112
Lucro (US$)	7.794.570	7.849.540	7.884.380	7.884.380	7.884.380	7.884.380	7.884.380

Tabela 4.5: Atribuição de frequência para cada cenário com 10 execuções independentes e 2000 iterações.

Destino	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
SBGR	22	21	21	21	21	21	21
SBSP	21	21	21	21	21	21	21
SBRJ	14	14	14	14	14	14	14
SBCF	14	14	14	14	14	14	14
SBKP	7	7	7	7	7	7	7
SBGL	7	7	7	7	7	7	7
SBPA	7	7	7	7	7	7	7
SBCY	7	7	7	7	7	7	7
SBPJ	7	7	7	7	7	7	7
SBCT	7	7	7	7	7	7	7
Total	113	112	112	112	112	112	112
Lucro (US$)	7.794.570	7.849.540	7.884.380	7.884.380	7.884.380	7.884.380	7.884.380

Tabela 4.6: Tempo de execução médio de cada cenário para CPU e GPU com 5 execuções independentes e 5000 iterações comparado a literatura.

Processamento	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
CPU	86.6089s	85.2627s	88.7055s	89.3581s	86.7590s	86.7361s	87.2070s
GPU	72.7365s	71.2787s	71.1288s	72.7179s	73.3854s	72.9893s	71.9716s
Literatura	136s	137s	78s	7s	6s	6s	6s

Tabela 4.7: Tempo de execução médio de cada cenário para CPU e GPU com 10 execuções independentes e 2000 iterações comparado a literatura.

Processamento	**1º**	**2º**	**3º**	**4º**	**5º**	**6º**	**7º**
CPU	32.3549s	33.9564s	33.6644s	28.1919s	27.7463s	27.6292s	27.9121s
GPU	28.0678s	29.0688s	28.4229s	34.8877s	33.3375s	33.5872s	33.5368s
Literatura	136s	137s	78s	7s	6s	6s	6s

4.5.1 Discussão

A frequência de voos considerada nas Tabelas 4.4 e 4.5 exibe apenas os voos de ida. Portanto, a quantidade real de voos deve levar em consideração os voos de volta. Por

Tabela 4.8: Comparação dos resultados de lucro obtidos para os cenários de 1 a 3 em U$.

Trabalho	**1º**	**2º**	**3º**
Jesus, Nagano e Celestino (2023)	U$7.855.657	U$8.130.775	U$8.134.799
Este trabalho	U$7.794.570	U$7.849.540	U$7.884.380
Déficit	0,008%	0,035%	0,031%

Tabela 4.9: Comparação dos resultados de lucro obtidos para os cenários de 4 a 7 em U$.

Trabalho	**4º**	**5º**	**6º**	**7º**
Jesus, Nagano e Celestino (2023)	U$8.134.799	U$8.134.799	U$8.134.799	U$8.134.799
Este trabalho	U$7.884.380	U$7.884.380	U$7.884.380	U$7.884.380
Déficit	0,031%	0,031%	0,031%	0,031%

exemplo, para o destino SBGR foram realizados 22 voos de SBGO para SBGR e 22 voos de SBGR para SBGO, totalizando 44 voos. É possível observar que, com exceção do primeiro cenário, a frequência de voos realizada manteve uma média de **112** voos, consistentemente alocando 21 voos para SBGR e SBSP. Jesus, Nagano e Celestino (2023) em seus resultados demonstram comportamento similar com relação à alta frequência de voos nestes dois destinos, por possuírem a maior demanda por turno da instância, com SBGR possuindo uma demanda de 134 passageiros e SBSP possuindo a demanda de 124 passageiros. Para os demais voos, as alocações mantiveram consistência em termos de baixa alocação, por serem voos de baixa demanda por turno. Nota-se que, seja com um grande número de iterações 4.4 4.2 ou com um baixo número de iterações 4.5 4.3 o algoritmo encontrou consistência nos resultados, obtendo o mesmo número de frequência e quantidade de aeronaves nos dois experimentos, o que é um comportamento esperado pela modelagem, conforme referenciado na literatura. No primeiro cenário, o modelo escolhido como mais lucrativo foi o A220-100, como é possível observar nas Tabelas 4.2 e 4.3. Este resultado se dá por diversos fatores:

- **Preço da passagem por passageiro**: O modelo A220-100 empata em termos de preço da passagem para cada destino com os modelos E190-E2 e ATR-72, modelos que nos cenários seguintes entram na solução, mostrando o potencial de cada um destes.

- **Custo operacional**: Quando comparado ao modelo E190-E2, o somatório do **CASK** para cada destino é menor para o modelo A220-100, além da capacidade de

passageiros do E190-E2 ser menor, fazendo com que ambos sejam candidatos para a seleção.

- **Estocasticidade do operador de mutação**: O operador de mutação 3.2.1 para o cenário 1 se comporta de maneira estocástica, alocando pelo menos 20% da demanda de passageiros para a construção da viagem. Nesta configuração, os voos com baixa quantidade de passageiros desempenha o papel fundamental na definição do modelo a ser utilizado, uma vez que é necessário balancear os custos a receita obtida.

Para os demais cenários, especialmente a partir do terceiro, os modelos utilizados rotacionam entre ATR-72, E190-E2 e A220-100, com o modelo E190-E2 dominando em termos de quantidade.

As aptidões dos indivíduos avaliados na CPU e na GPU permaneceram idênticas, devido à operação de *static_cast* para o tipo *float* realizada na atribuição do dado de aptidão ao indivíduo no término do cálculo da função objetiva realizado na GPU, truncando casas decimais de precisão presentes na variável utilizada para o cálculo. Em comparação aos resultados obtidos por Jesus, Nagano e Celestino (2023), as Tabelas 4.8 e 4.9 mostram um comparativo entre o lucro obtido por este trabalho e o valor ótimo encontrado pelo artigo em cada cenário. Os autores utilizaram o algoritmo Coin-OR Branch-and-Cut (CBC) presente no pacote Python PuLP (MITCHELL et al., 2025) e, à medida que a frota disponível pode ser mais diversa, a partir do cenário 3, o algoritmo encontra uma solução e a repete para os cenários seguintes, característica do modelo de ser menos restrito com uma frota mais diversa, facilitando o algoritmo a encontrar uma solução.

Tempo de Execução

O tempo de execução médio de uma solução, quando comparados os tempos da CPU e GPU em um mesmo cenário para o primeiro experimento, evidencia um ganho médio de ~ 15 segundos de tempo, mostrando uma melhoria na velocidade de obtenção da solução. Para o segundo conjunto de experimentos, o ganho é de $\sim 0,912$ segundos, evidenciando que mesmo com um menor número de iterações, existe um ganho de desempenho ao utilizar o paralelismo. Para o primeiro, segundo e terceiro cenário, obtivemos uma solução mais rápida, com o tempo necessário para obtê-la pelos autores de 136 segundos para o

primeiro cenário, 137 segundos para o segundo e 78 segundos para o terceiro, mostrando uma diminuição no tempo de 46% para o primeiro cenário no primeiro conjunto de experimentos 4.6, 79% para o segundo conjunto de experimentos 4.7; no segundo cenário, uma redução de 48% para o primeiro conjunto e 79% para o segundo conjunto; no terceiro cenário, uma redução de 0,06% para o primeiro conjunto e 56% para o segundo conjunto. Para os demais cenários, com relação ao tempo de execução da literatura, os autores afirmam que o modelo se torna mais fácil de resolver à medida que mais modelos de aviões estão disponíveis para uso. O AG mantém a mesma média de tempo para solucionar todos os cenários, devido ao critério de parada ser somente o número de iterações.

5 Conclusão

A representação do cromossomo implementado no Algoritmo Genético, permitiu que as soluções obtidas fossem viáveis quanto às restrições de fluxo, demanda e tipos de aeronaves disponíveis do modelo adotado, garantindo que qualquer solução intermediária da busca fosse viável, mesmo não havendo boa qualidade.

A partir dos resultados obtidos nos experimentos, é evidente a consistência na convergência das soluções em todos os cenários simulados, com o lucro total permanecendo similar a partir do terceiro cenário, comportamento que se manteve mesmo com um alto número de iterações para o algoritmo. O operador de crossover, para possibilitar a troca parcial entre a programação dos indivíduos pais, necessita de um aumento na complexidade da implementação, levando em consideração a frota disponível e trocas que possibilitem um ganho positivo de aptidão do indivíduo. A heurística simples empregada no operador de mutação cria uma tendência de preencher a capacidade do modelo de avião utilizado a partir do segundo cenário, convergindo no máximo local de U$7.884.380 partindo do terceiro cenário.

Para trabalhos futuros, um estudo na especialização dos operadores genéticos, principalmente no crossover, é de grande interesse, possibilitando novas soluções e expandindo a capacidade exploratória do algoritmo. Para o operador de mutação, uma especialização similar ao caso do primeiro cenário, pode ser benéfica para uma maior variedade de alocações de passageiros, favorecendo outros modelos para compor a frota. Um algoritmo construtivo mais especializado para a população inicial poderia acelerar a convergência dos resultados, quando comparado à inicialização utilizando mutações. O uso de estratégias de busca local para aprimorar as soluções obtidas, considerando a natureza do modelo de convergir em um resultado ótimo, se demonstra uma oportunidade de aprimoramento geral do resultado do algoritmo. A utilização da programação paralela, particularmente utilizando CUDA, proporcionou um ganho de desempenho e velocidade de obtenção da solução, especialmente quando a quantidade de iterações para o algoritmo é maior, evidenciando a diferença de tempo. É importante ressaltar que para a

implementação em GPU demonstrar ganhos sobre a implementação sequencial em CPU, é necessário que a quantidade de indivíduos na população, bem como o tamanho de suas soluções, justifique a utilização do paralelismo. O tempo de transferência de dados da população entre a GPU e a CPU pode não justificar o paralelismo, com a implementação em sequencial sendo mais vantajosa por não necessitar do passo de transferência e de sincronia. Para trabalhos futuros, uma maior paralelização das etapas da busca na GPU, implementando outras estratégias como MSGAs assíncronas, em rede ou distribuída, explorando o potencial do hardware da GPU, eliminando a etapa de transferência de dados e diminuindo o tempo para obter soluções.

Bibliografia

ARORA, R.; TULSHYAN, R.; DEB, K. Parallelization of binary and real-coded genetic algorithms on gpu using cuda. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8.

ARORA, R.; TULSHYAN, R.; DEB, K. Parallelization of binary and real-coded genetic algorithms on gpu using cuda. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8.

BARNHART, C. et al. Flight string models for aircraft fleeting and routing. *Transportation Science*, INFORMS, v. 32, n. 3, p. 208–220, 1998. ISSN 00411655, 15265447.

Ben Ahmed, M. et al. A matheuristic for the robust integrated airline fleet assignment, aircraft routing, and crew pairing problem. *Computers & Operations Research*, v. 137, p. 105551, 2022. ISSN 0305-0548.

CANTú-PAZ, E. *Master-Slave Parallel Genetic Algorithms*. Boston, MA: Springer US, 2001. 33–48 p. ISBN 978-1-4615-4369-5.

COOK, G.; GOODWIN, J. Airline networks: A comparison of hub-and-spoke and point-to-point systems. *Journal of Aviation/Aerospace Education; Research*, v. 17, n. 2, 2008.

DEGASPERI, P. *Demanda mundial por transporte aéreo de carga aumenta 11,4% em agosto*. 2024. ⟨https://institutoaviacao.org/noticias/demanda-mundial-por-transporte-aereo-de-carga-aumenta-114-em-agosto/⟩.

DEVECI, M.; DEMIREL, N. Çetin. Evolutionary algorithms for solving the airline crew pairing problem. *Computers & Industrial Engineering*, v. 115, p. 389–406, 2018. ISSN 0360-8352.

GHAZFAN, D.; BALA, S.; NOLAN, M. Massively parallel genetic algorithms. 01 1994.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

HE, Y. et al. Maximizing robustness of aircraft routing with heterogeneous maintenance tasks. *Transportation Research Part E: Logistics and Transportation Review*, v. 177, p. 103237, 2023. ISSN 1366-5545.

JESUS, T. D. de; NAGANO, M. S.; CELESTINO, V. R. R. An integrated frequency assignment and fleet assignment model: A strategic application in the brazilian regional aviation market. *Transport Economics and Management*, v. 1, p. 151–159, 2023. ISSN 2949-8996.

KENAN, N.; JEBALI, A.; DIABAT, A. The integrated aircraft routing problem with optional flights and delay considerations. *Transportation Research Part E: Logistics and Transportation Review*, v. 118, p. 355–375, 2018. ISSN 1366-5545.

KHANMIRZA, E.; NAZARAHARI, M.; HAGHBEIGI, M. A heuristic approach for optimal integrated airline schedule design and fleet assignment with demand recapture. *Applied Soft Computing*, v. 96, p. 106681, 2020. ISSN 1568-4946.

MAZZA, K. *Aviação comercial brasileira contribuiu com R$ 81 bilhões para o PIB em 2023*. 2024. ⟨https://institutoaviacao.org/noticias/aviacao-comercial-brasileira-contribuiu-com-r-81-bilhoes-para-o-pib-em-2023/⟩.

MAZZA, K. *Com 118 milhões de passageiros transportados em 2024, setor aéreo tem segundo melhor desempenho da história*. 2025. ⟨https://institutoaviacao.org/noticias/com-118-milhoes-de-passageiros-transportados-em-2024-setor-aereo-tem-segundo-melhor-desempen⟩.

MITCHELL, S. et al. *Optimization with pulp*. 2025. Disponível em: ⟨https://coin-or.github.io/pulp/⟩.

NVIDIA; VINGELMANN, P.; FITZEK, F. H. *CUDA, release: 10.2.89*. 2020. Disponível em: ⟨https://developer.nvidia.com/cuda-toolkit⟩.

OUYANG, W.; ZHU, X. Meta-heuristic solver with parallel genetic algorithm framework in airline crew scheduling. *Sustainability*, v. 15, n. 2, 2023. ISSN 2071-1050.

PETROWSKI, A.; BEN-HAMIDA, S. *Evolutionary Algorithms*. Wiley, 2017. (Computer Engineering: Metaheuristics). ISBN 9781848218048. Disponível em: ⟨https://books.google.com.br/books?id=fvRRCgAAQBAJ⟩.

RUTHER, S. et al. Integrated aircraft routing, crew pairing, and tail assignment: Branch-and-price with many pricing problems. *Transportation Science*, v. 51, n. 1, p. 177–195, 2017.

TALBI, E.-G. *Metaheuristics: From Design to Implementation*. [S.l.]: Wiley Publishing, 2009. ISBN 0470278587.

XU, Y.; WANDELT, S.; SUN, X. Airline integrated robust scheduling with a variable neighborhood search based heuristic. *Transportation Research Part B: Methodological*, v. 149, p. 181–203, 2021. ISSN 0191-2615.

ZEREN, B.; ÖZCAN, E.; DEVECI, M. An adaptive greedy heuristic for large scale airline crew pairing problems. *Journal of Air Transport Management*, v. 114, p. 102492, 2024. ISSN 0969-6997.