ALGORITMO GENÉTICO PARA O PROBLEMA DO POUSO DE AERONAVES EM ÚNICA PISTA

Trabalho de Conclusão de Curso

Engenharia da Computação

Nilton Vieira da Silva
Orientador: Prof. Eliane Maria Loiola

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

NILTON VIEIRA DA SILVA

ALGORITMO GENÉTICO PARA O PROBLEMA DO POUSO DE AERONAVES EM ÚNICA PISTA

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, Outubro de 2022.

Silva, Nilton Vieira da

Algoritmo Genético para o Problema de Pouso de Aeronaves em Única Pista / Nilton Vieira da Silva. – Recife - PE, 2022.

xiii, 55 f. : il. ; 29 cm.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) Universidade de Pernambuco, Escola Politécnica de Pernambuco, Recife, 2021.

Orientador: Profª. Drª. Eliane Maria Loiola.

Inclui referências.

1. Otimização combinatória. 2. Meta-heurística Algoritmo Genético. 3. Problema de Pouso de Aeronaves. I. Algoritmo Genético para o Problema de Pouso de Aeronaves em Única Pista. II. Loiola, Eliane Maria. III. Universidade de Pernambuco.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 24/10/2022, às 16h00min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **NILTON VIEIRA DA SILVA**, orientado(a) pelo(a) professor(a) **ELIANE MARIA LOIOLA**, sob título Algoritmo Genético para o Problema do Pouso de Aeronaves em única Pista, a banca composta pelos professores:

CARMELO JOSE ALBANEZ BASTOS FILHO (PRESIDENTE)
ELIANE MARIA LOIOLA (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 9,00 (nove)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

AVALIADOR 1: Prof (a) **CARMELO JOSE ALBANEZ BASTOS FILHO**

AVALIADOR 2: Prof (a) **ELIANE MARIA LOIOLA**

AVALIADOR 3: Prof (a)

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Dedico este trabalho a meus pais,
por todo o apoio e carinho,
Obrigado.

Agradecimentos

Gostaria de agradecer também a toda minha família e amigos e em especial aos meus pais, pelo grande apoio e investimento em todos os momentos motivadores para a minha graduação.

Gostaria de agradecer também a todos os grandes amigos e agora companheiros de trabalho que conheci durante minha graduação, sem os quais esta jornada acadêmica jamais poderia ser tão memorável, em especial a Matheus Phelipe, Murilo Stodolni, Richard Jeremias, Pedro Cunha, Michael Matheus, Matheus Albert e Selton Guedes.

Além disso gostaria de agradecer a todo o corpo docente do ECOMP, por ajudar na construção de todo o conhecimento adquirido durante a graduação e em especial a Professora Eliane Maria Loiola por me orientar neste trabalho.

Muito obrigado! <3

ESCOLA POLITÉCNICA
DE PERNAMBUCO

Autorização de publicação de PFC

Eu, **Nilton Vieira da Silva** autor(a) do projeto de final de curso intitulado: **Algoritmo Genético para o Problema do Pouso de Aeronaves em única Pista;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.

Nilton Vieira da Silva

Nilton Vieira da Silva

Eliane

Orientador(a): **Eliane Maria Loiola**

Coorientador(a):

Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 24/10/2022

Resumo

Algoritmos genéticos são meta-heurísticas que se inspiram na teoria evolução das espécies e da seleção natural para a resolução de problemas complexos, como problemas de otimização combinatória. O presente trabalho traz uma implementação de um algoritmo genético voltado ao problema de pouso de aeronaves, que busca determinar o melhor sequenciamento para a realização da operação de pousos de aeronaves em uma pista de operações, com o objetivo de encontrar uma solução com custo próximo ao custo ótimo, entre as soluções viáveis do problema, para isso será apresentado um experimento utilizando um conjunto de instâncias do problema para um caso estático disponíveis na OR-Library. Por meio dos ciclos de experimento foram encontradas soluções viáveis em tempo não proibitivo e de custos similares, quando não melhores do que resultados obtidos por outros algoritmos implementados para o mesmo problema, os resultados foram analisados de forma a verificar a evolução do custo de cada solução durante a progressão do algoritmo e a efetividade e o impacto dos parâmetros utilizados para a exploração do espaço de busca.

Palavras-chave: Algoritmo Genético, Meta-heurística, Otimização Combinatória, Problema de Pouso de Aeronaves.

Abstract

Genetic algorithms are meta-heuristics that are inspired by the theory of evolution of species and natural selection to solve complex problems, such as combinatorial optimization problems. The present work presents an implementation of a genetic algorithm aimed at the aircraft landing problem, which seeks to determine the best sequencing to perform the operation of aircraft landings on a runway, with the objective of finding a solution with a cost close to the optimal cost, among the viable solutions of the problem, for this an experiment will be presented using a set of problem instances for a static case available in the OR-Library. Through the experiment cycles, viable solutions were found in a non-prohibitive time and with similar costs, when not better than results obtained by other algorithms implemented for the same problem, the results were analyzed in order to verify the evolution of the cost of each solution during the progression of the algorithm and the effectiveness and impact of the parameters used for the exploration of the search space.

Keywords: Genetic Algorithm, Meta-heuristic, Combinatorial Optimization, Aircraft Landing Problem.

Lista de Ilustrações

Lista de Tabelas

Lista de Símbolos e/ou Siglas

ACO	*Ant Colony Optimization*
ALP	*Aircraft Landing Problem*
FCFS	*First-Comes, First Served*
FIFO	*First-In First-Out*
GA	*Genetic Algorithm*
NP	*Nondeterministic Polynomial*

Sumário

1 Introdução

Grande parte dos problemas práticos que possuem um número finito ou infinito de soluções alternativas podem ser formulados como um problema de otimização combinatória (OSMAN, 1995). Esta otimização pode ser definida como o estudo matemático de encontrar um arranjo ótimo, um agrupamento, uma ordenação ou uma seleção de objetos discretos geralmente finitos em números (LAWLER, 1976), de forma a alcançar um resultado ótimo (máximo ou mínimo) respeitando as restrições intrínsecas ao problema, comumente este resultado é medido através de uma função objetivo.

De uma forma intuitiva, para conseguir e garantir que um determinado arranjo de solução é ótimo, seria necessário analisar todas as combinações possíveis, de modo a comparar o resultado da função objetivo de cada uma dessas possibilidades, esta estratégia é conhecida como método de força bruta ou busca exaustiva. Contudo, dependendo do tamanho do problema, a estratégia de força bruta torna-se inviável, pois em situações em que o tempo e/ou custo de processamento é importante, não se consegue chegar a uma solução ótima em um tempo razoável. Isso é explicado pela teoria da complexidade computacional descoberta por Cook (1971), que tenta categorizar o requisito computacional dos algoritmos e os problemas práticos encontrados como “fácil” ou “difícil”. Garey e Johnson (1979) explicam a noção de tempo razoável de processamento e demonstram que para muitos problemas combinatórios não existem um algoritmo conhecido que realize a busca completa no espaço de solução em tempo polinomial, sendo alguns destes problemas classificados e conhecidos atualmente como NP-difíceis.

A busca exaustiva de uma solução ótima para problemas de otimização combinatória torna-se inviável quando aumentado o tamanho do problema (número de variáveis envolvidas), dada a quantidade de combinações possíveis entre seus elementos (CUNHA *et al.*, 2012). O uso de métodos exatos é inviável para resolver instâncias reais de alguns problemas de otimização combinatória, em virtude do seu tamanho e, torna-se evidente a necessidade de adotar o uso de outras técnicas.

Uma técnica que vem sendo amplamente utilizada é o uso de métodos aproximativos para explorar o espaço de solução em busca de bons resultados, vasculhando-o de forma inteligente, sem comprometer a eficiência computacional, mas ao preço de não garantir encontrar a solução ótima (BLUM; ROLI, 2003).

Neste contexto, este trabalho explora a versatilidade dos algoritmos aproximativos através de uma abordagem experimental como uma alternativa de solução para o problema de pouso de aeronaves (do inglês, *ALP – Aircraft Landing Problem*), causado pela dificuldade de fazer um planejamento adequado no sentido de implementar uma utilização eficiente do uso das pistas onde ocorre pouso e decolagem das aeronaves. A complexidade computacional do problema com pista única mostrou ser do tipo NP-Difícil, visto que além da complexidade computacional inerente ao problema, a magnitude da dificuldade do problema é exacerbada pela consideração de incertezas e múltiplos objetivos conflitantes (SOYKAN, 2016). O que significa que não há algoritmos conhecidos para encontrar soluções ótimas em tempo polinomial para problemas reais, em virtude de suas dimensões.

1.1. Justificativa

Por se tratar de um problema de otimização combinatória, Segundo Beasley *et al.* (2000), a resolução de instâncias do ALP exige grandes esforços computacionais, que crescem exponencialmente com o tamanho do problema, mesmo para cenários estáticos, por se tratar de um problema NP-Difícil. Neste sentido, em um sequenciamento simples de 5 aeronaves dentro de um intervalo de 500 unidades de tempo é possível alcançar um domínio de mais de 31 bilhões de soluções, sendo elas viáveis ou não.

Sendo assim, o presente trabalho busca contribuir para o uso de algoritmos genéticos aplicados à área da aviação, visto ser um algoritmo bastante utilizado em problemas de otimização combinatória, com ênfase em otimizar o agendamento de aeronaves em uma pista de operações visando minimizar os custos relacionados aos atrasos e adiantamento das aeronaves para realização do pouso e decolagem.

1.2. Objetivos

O objetivo geral deste trabalho é verificar a viabilidade, eficiência e aplicabilidade do uso da meta-heurística algoritmo genético para o problema de pouso de aeronaves.

1.2.1. Objetivos específicos

Para alcançar o objetivo geral, este trabalho possui os seguintes objetivos específicos:

- Compreender e descrever o modelo matemático para solucionar o problema do pouso de aeronaves em múltiplas pistas;
- Realizar um entendimento acerca dos algoritmos genéticos e suas formas de implementação;
- Propor adaptações no algoritmo para torná-lo mais eficiente e adequado ao cenário em questão.

1.3. Metodologia

Esta pesquisa tem natureza aplicada, exploratória e quantitativa (BARROS NETO, 2002). Tem como objetivo desenvolver conhecimentos na resolução de problemas computacionais concretos, explorando diversos conceitos sobre o tema de pesquisa fazendo uma análise quantitativa dos resultados encontrados através dos experimentos realizados.

Para isso, neste trabalho, serão discutidos os conceitos envolvendo algoritmos genéticos, será apresentada uma formulação do ALP considerando o cenário estático, onde todas as informações são conhecidas, por fim, será apresentado um algoritmo genético adaptado para obter uma solução do problema alvo.

1.4. Estrutura do trabalho

Além deste capítulo introdutório, este trabalho se encontra estruturado da seguinte forma: o Capítulo 2 descreve a importância dos algoritmos aproximativos como método de resolução de problemas de otimização combinatória, apresentando mais detalhadamente a meta-heurística algoritmo genético; o Capítulo 3 apresenta a formulação do problema de pouso de aeronaves e um algoritmo genético adaptado para este problema; o Capítulo 4 apresenta o experimento realizado nesta pesquisa e os resultados encontrados; por fim, o Capítulo 5 discute algumas considerações finais e trabalhos futuros.

2 Meta-heurística

Heurísticas e meta-heurísticas são uma classe de métodos aproximados desenvolvidos no início dos anos 80 (OSMAN; LAPORTE, 1996). Elas são voltadas para problemas de otimização combinatória complexos, onde as heurísticas clássicas e métodos de otimização tradicionais têm falhado em serem efetivos e eficientes. Uma heurística é definida como um procedimento proposto para resolver problemas utilizando-se da estrutura do problema de forma a ser interpretado ou explorado inteligentemente para obter uma solução razoável (NICHOLSON, 1971). E, as meta-heurísticas são heurísticas de propósito geral que combinam de forma inteligente diferentes conceitos para explorar o espaço de busca, utilizando estratégias de aprendizagem usadas como uma estrutura de informação para encontrar boas soluções com baixo custo computacional (SOYKAN, 2016).

As meta-heurísticas se utilizam de técnicas que são baseadas em conceitos já conhecidos como: evolução biológica dos seres vivos, sistema nervoso, mecanismos estatísticos, entre outros, de forma a utilizar esses conceitos pré-estabelecidos para resolver problemas complexos, inclusive problemas de otimização. Exemplos de meta-heurísticas clássicas são: algoritmos genéticos, algoritmos de colônias de formigas, recozimento simulado, busca tabu, e outros vários (OSMAN; LAPORTE, 1996).

As soluções encontradas por uma determinada meta-heurística não são garantidamente soluções ótimas e, em geral, as meta-heurísticas não são capazes de verificar o quão perto estão da solução ótima. Ainda assim, estas técnicas vêm sendo aprimoradas de forma que conseguem se aproximar cada vez mais de propriedades de convergências ótimas e produzir soluções de alta qualidade (próximas do ótimo) através de estruturas de memórias e formas adaptativas de aprendizagem (OSMAN, 1995).

Além do fato de serem mais amigáveis quanto a complexidade computacional, Silver *et al.* (1980) cita outras vantagens em se utilizar métodos meta-heurísticos, como, por exemplo, o fato de serem algoritmos mais flexíveis, generalistas, requerem menos esforço para serem implementados, alcançarem soluções quase

ótimas mais rapidamente, quando comparados a outros métodos tradicionais, além de serem de fácil modificação, mas, apesar de tudo isso, não existe um único algoritmo que seja adequado para todos os problemas de otimização (WOLPERT; MACREADY, 1997).

De uma forma geral, meta-heurísticas são mais adequadas para problemas mais vagos, onde até mesmo o modelo matemático do problema seria desconhecido, ou para os casos nos quais é inviável aplicar métodos de resolução tradicionais, devido ao alto custo computacional exigido para instâncias de tamanhos do mundo real, que é o caso do problema de escalonamento de pouso e decolagem de aeronaves.

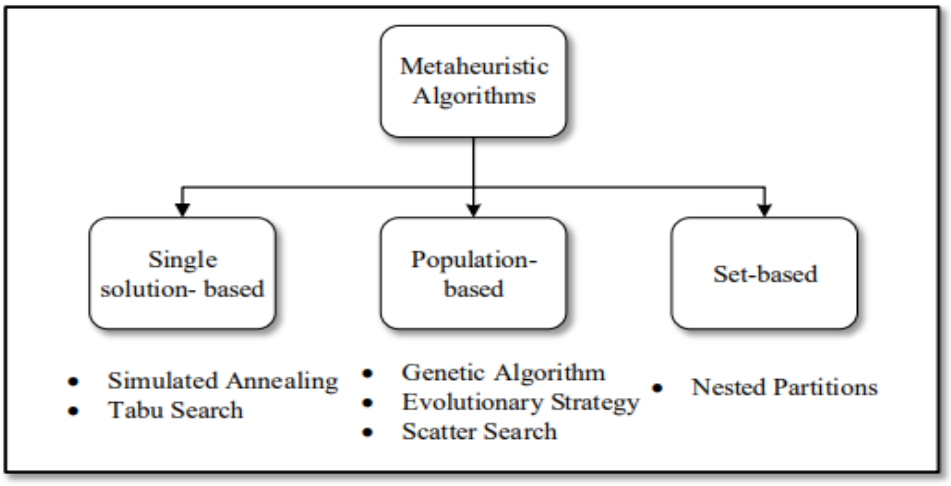
2.1. Categorias de meta-heurísticas

Algoritmos meta-heurísticos podem ser categorizados de várias formas, mas a maneira mais comum de serem categorizados é baseado no número de soluções candidatas mantidas e melhoradas de forma simultânea (SOYKAN, 2016). Estes problemas podem ser divididos em três grupos: solução única, populacional e baseada em conjuntos, conforme apresentado na Figura 1.

Nos algoritmos baseados em solução única, também conhecidos como algoritmos baseados em trajetória, um único estado é preservado durante o processo de otimização, e o processo de busca é utilizado para melhorar este único estado. Já algoritmos populacionais possuem uma população de soluções candidatas guardadas e atualizadas a cada iteração, em vez de seguir um único caminho no espaço de busca.

As meta-heurísticas baseadas em conjuntos usa uma estratégia de amostragem global que é continuamente adaptada usando o particionamento do espaço de busca em conjuntos (SOYKAN, 2016).

Figura 1 – Classificação das meta-heurísticas.



Fonte: SOYKAN (2016).

O projeto de uma meta-heurística tenta conciliar dois pontos fundamentais, e contraditórios, do comportamento desses algoritmos que guiam a forma de como irão executar a busca por sua solução. O primeiro deles é o mecanismo de intensificação que é voltado para explorar as áreas mais promissoras do problema na esperança de encontrar soluções ainda melhores, e é comumente implementado através de técnicas de busca local. O outro é o mecanismo de diversificação que é voltado a ampliar as possibilidades do processo de busca para se mover em busca de novas áreas ainda não exploradas (TALBI, 2009).

Este trabalho explora a potencialidade dos algoritmos baseados em população, mais precisamente algoritmos genéticos. E, na próxima seção, serão apresentados os conceitos fundamentais envolvendo algoritmos genéticos.

2.2. Algoritmo genético

No século XIX, J. Mendel foi o primeiro a estabelecer as bases da hereditariedade de pais para seus descendentes. Logo depois, Darwin (1859) apresentou a teoria da evolução presente no seu livro A Origem das Espécies. Longo tempo depois, na década de 1970, essas teorias de criação de novas espécies e sua evolução inspirou diversos pesquisadores a projetar algoritmos evolutivos.

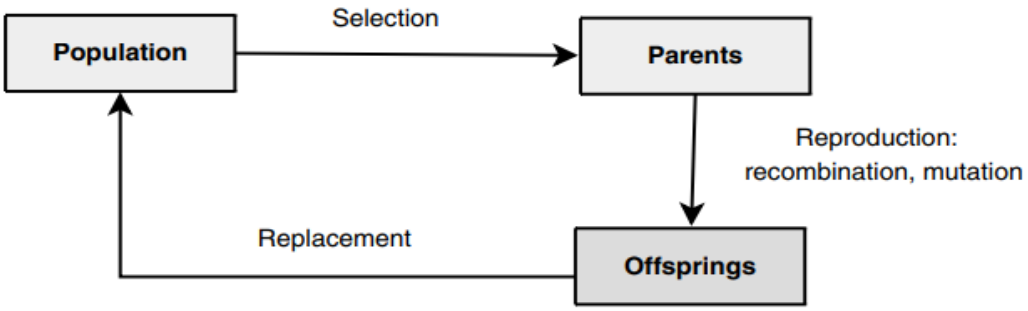
Algoritmos evolucionários são meta-heurísticas populacionais estocásticas que vem tendo sucesso em diversas aplicações de problemas complexos (multi-modal, multi-objetivo, e problemas de múltiplas restrições), como também em resolver problemas de otimização em vários domínios (otimização contínua ou combinatória, modelagem e identificação de sistemas, mineração de dados e aprendizado de máquina), o que fez deles os algoritmos populacionais mais estudados e promoveu o campo conhecido como computação evolucionária composto inicialmente por algoritmos genéticos, estratégia evolutiva e programação evolucionária, entre outros (TALBI, 2009).

Algoritmos genéticos pertencem a classe de métodos evolutivos que adotam a teoria da evolução da variação genética e seleção natural, onde indivíduos mais adaptados tem uma maior chance de se reproduzir e deixar seus descendentes para a próxima geração (SOYKAN, 2016). Inspirados por esses princípios, soluções de baixa qualidade são eliminadas da população, e os indivíduos mais aptos se reproduzem para garantir descendentes bem-sucedidos, mantendo uma população de soluções candidatas que evolua sob uma pressão seletiva que favoreça as melhores soluções.

Isso significa que os genes de indivíduos mais aptos se espalharão para um número crescente de indivíduos em cada geração sucessiva. A combinação de boas características de pais altamente adaptados pode produzir descendentes ainda mais aptos. Desta forma, as espécies evoluem para se adaptarem cada vez melhor ao seu ambiente, resultando em melhores soluções (BEASLEY *et al.*, 2001).

Algoritmos genéticos são procedimentos iterativos que opera em uma população finita de n cromossomos (soluções), que é análogo ao material genético de cada organismo. Em seguida, essa população evolui ao longo de várias iterações a partir da seleção de indivíduos e do uso de operadores genéticos como cruzamento e mutação, onde cada iteração é chamada de “geração”, que resulta em descendentes herdando propriedades de seus pais. Os descendentes são avaliados de acordo com alguma função de aptidão que avalia a habilidade de cada indivíduo de sobreviver no ambiente do problema estudado, os indivíduos com mais habilidades são colocados na população, possivelmente substituindo os membros mais fracos da última geração (OSMAN, 1995).

Figura 2 – Mecanismo de busca do algoritmo genético.



Fonte: TALBI (2009).

Assim, o mecanismo de busca consiste em três fases como pode ser visto na Figura 2: avaliação da aptidão de cada cromossomo, seleção dos cromossomos pais e aplicação dos operadores de mutação e recombinação aos cromossomos pais. Os novos cromossomos resultantes dessas operações formam a população para a próxima geração e o processo se repete até que o sistema pare de melhorar. A sobrevivência dos indivíduos mais aptos garante que a qualidade geral das soluções aumente à medida que o algoritmo progride de uma geração para a próxima.

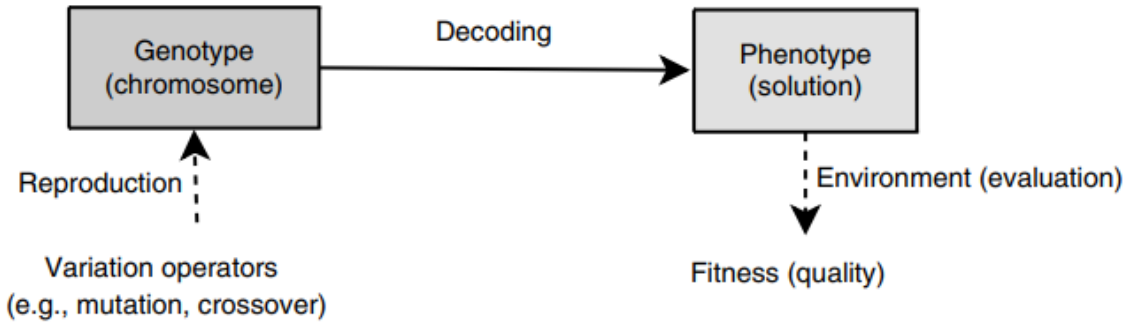
2.2.1. Representação do indivíduo

O primeiro algoritmo genético foi desenvolvido por Holland (1975), através de estudos de autômatos celulares conduzido por ele e seus colegas. Algoritmos genéticos tem esse nome por enfatizar a representação e manipulação dos indivíduos em termos de sua composição genética em vez de usar uma representação fenotípica (OSMAN, 1995). Em sua proposta, Holland (1975) teve a ideia de representar uma possível solução para o problema utilizando codificação binária que é considerada a base matemática dos algoritmos genéticos.

Nos algoritmos evolucionários de uma forma geral, o genótipo representa a codificação enquanto o fenótipo representa a solução. Logo, os genótipos precisam ser decodificados para gerar fenótipos. Os operadores genéticos atuam no nível dos

genes, enquanto a avaliação do indivíduo feita através da função objetivo usam do fenótipo associado a cada indivíduo (TALBI, 2009).

Figura 3 – Genótipo e fenótipo no algoritmo genético.



Fonte: (TALBI, 2009).

Sendo assim, nos casos que utilizam codificação direta, o genótipo é similar ao fenótipo, porém nos casos de codificação indireta os genótipos e os fenótipos possuem diferentes estruturas e, portanto, torna-se necessário o uso de uma função de decodificação para transformar um dado genótipo em fenótipo, como pode ser visto na Figura 3.

2.2.2. População inicial

Nos algoritmos genéticos, existem diversas possibilidades de se criar uma população inicial e não existe uma regra que defina uma melhor inicialização, em muito dos casos, a população inicial é gerada aleatoriamente, contudo também podem ser utilizadas soluções heurísticas que possam trazer uma melhor qualidade como mostrado na Tabela 1 (OSMAN, 1995).

Tabela 1 – Estratégias para construir uma população inicial.

Strategy	Diversity	Computational Cost	Quality of Initial Solutions
Pseudo-random	++	+++	+
Quasi-random	+++	+++	+
Sequential diversification	++++	++	+
Parallel diversification	++++	+++	+
Heuristic	+	+	+++

Fonte: TALBI (2009).

Assim como a forma, o tamanho da população também é um parâmetro fundamental a ser considerado. Uma população muito grande pode se tornar computacionalmente inviável devido ao aumento da complexidade computacional, assim como ter uma população muito pequena pode causar uma taxa de convergência prematura ou perda de diversidade, que leva mais rapidamente a estacionar em soluções ótimas locais rapidamente (JONES, 1993).

2.2.3. Seleção de indivíduos

O mecanismo de seleção é um dos principais componentes de busca nos algoritmos evolucionários. Um princípio bastante utilizado pelos métodos de seleção é “quanto melhor é um indivíduo, maior é sua chance de ser um pai”. Tal pressão de seleção levará a população a melhores soluções. No entanto, os piores indivíduos não devem ser descartados imediatamente e por isso ainda restam chances de serem selecionados, podendo levar a um material genético útil.

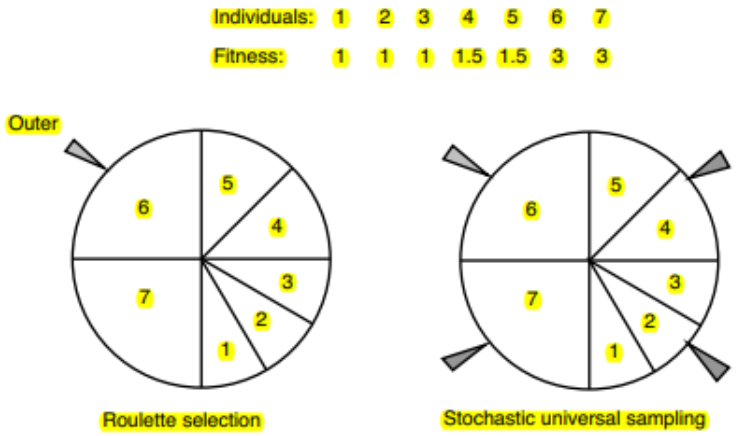
A estratégia adotada para seleção determina quais indivíduos são escolhidos para o acasalamento (reprodução) e quantos descendentes cada indivíduo selecionado produz. Nos algoritmos evolucionários, a seleção através do uso do parâmetro de desempenho pode ser feita de forma proporcional, onde o valor absoluto do desempenho é associado ao indivíduo, ou baseado em classificação, onde valores absolutos de desempenho são classificados e, posteriormente, associados ao indivíduo.

Em seguida, os pais são selecionados de acordo com sua aptidão por meio de alguma estratégia, TALBI (2009) aponta algumas mais utilizadas como: seleção

por roleta, amostragem estocástica universal, seleção por torneio e seleção baseada em classificação.

A seleção por roleta é a estratégia de seleção mais comum, onde atribui a cada indivíduo uma probabilidade de seleção proporcional ao seu desempenho, ou seja, quanto melhor adaptado ao problema maiores a chances de ser escolhido para reproduzir e passar seus genes para futuras gerações. Contudo, indivíduos destacados acabam introduzindo um viés no início da busca que pode causar uma convergência prematura e perda de diversidade, visto que indivíduos piores tem cada vez menos chances de se reproduzir a medida que soluções melhores são encontradas. Além disso, quando todos os indivíduos são igualmente aptos, essa estratégia de seleção não introduz pressão suficiente para selecionar os melhores indivíduos.

Figura 4 – Seleção de indivíduo por roleta e por amostragem.

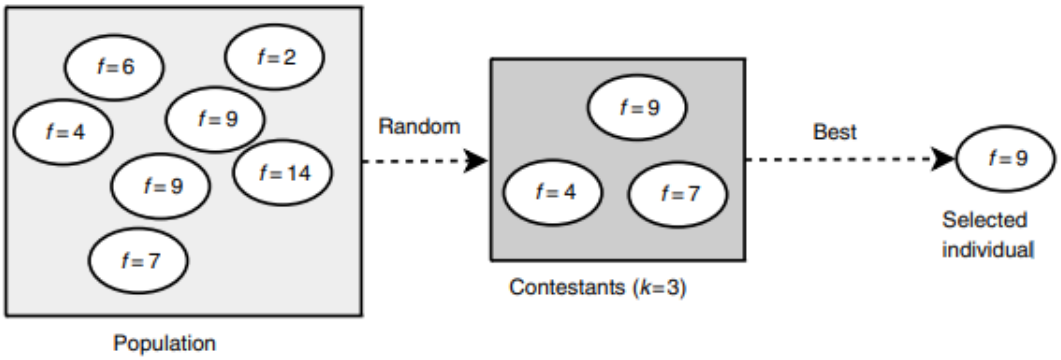


Fonte: TALBI (2009).

Visando diminuir o viés inerente à seleção por roleta, a amostragem universal estocástica pode ser utilizada, onde uma roleta externa é colocada ao redor da dos candidatos com k ponteiros igualmente espaçados. Nessa estratégia, um único giro da roleta selecionará simultaneamente todos os k indivíduos para reprodução. A Figura 4 ilustra os princípios da roleta e as estratégias de seleção.

A seleção por torneio consiste em selecionar aleatoriamente k indivíduos, onde o parâmetro k é chamado de tamanho do grupo do torneio. Um torneio é então aplicado aos k membros do grupo para selecionar o melhor. Para enfim selecionar p indivíduos, o procedimento de torneio é então realizado p vezes. Conforme ilustrado na Figura 5.

Figura 5 – Seleção de indivíduo por torneio.



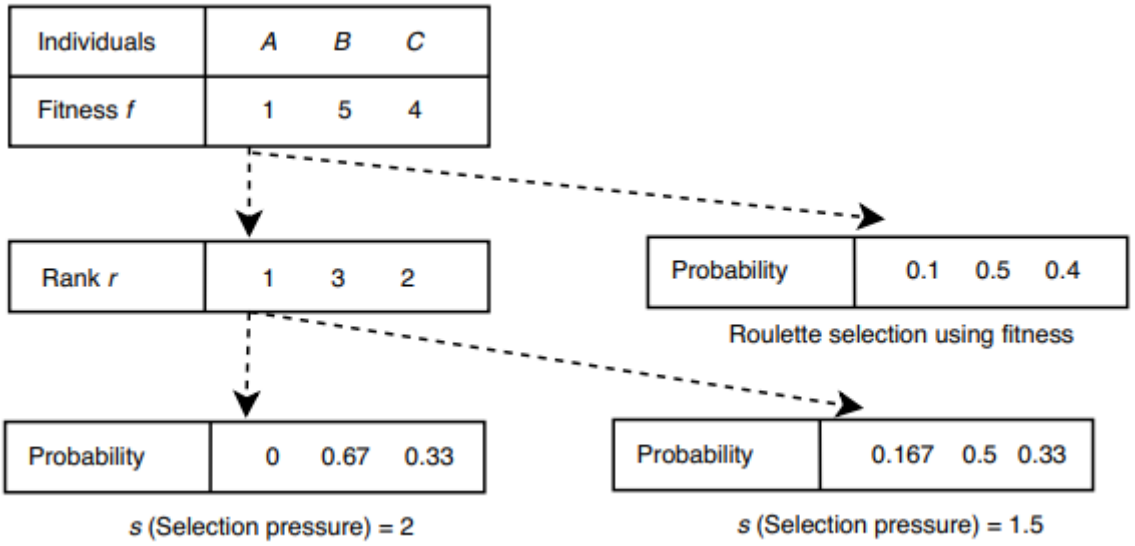
Fonte: TALBI (2009).

Em vez de usar o valor de desempenho de um indivíduo, a seleção baseada em classificação de indivíduos é usada. A função é tendenciosa para indivíduos com uma classificação alta (ou seja, boa aptidão). A classificação pode ser dimensionada linearmente usando a equação (1).

$$P(i) = \frac{2-s}{\mu} + \frac{2 \cdot r(i)(s-1)}{\mu\,(\mu-1)} \quad (1)$$

Onde s é a pressão de seleção (1,0 < s ≤ 2,0), μ é o tamanho da população e $r(i)$ é a classificação associado ao indivíduo *i*. Quanto maior é a pressão de seleção *s*, mais importância é dada aos melhores indivíduos, como pode ser visto na Figura 6.

Figura 6 – Seleção de indivíduo por classificação.



Fonte: TALBI (2009).

2.2.4. Mecanismo de reprodução

Após a etapa de seleção dos indivíduos, inicia-se o processo de reprodução que é responsável por guiar o algoritmo em busca das melhores soluções utilizando de mecanismos como mutação e cruzamento, em prol de percorrer o espaço de busca visando conseguir melhores soluções.

O operador de mutação é um operador unário que atuam em um único indivíduo, onde irá representar pequenas mudanças de indivíduos selecionados da população (TALBI, 2009). O operador de mutação é fundamental importância afim de ampliar e diversificar o espaço de busca em prol de encontrar melhores soluções, ele atuará através de uma probabilidade que define a chance de mutar cada elemento (gene) da representação. Em geral, é recomendado utilizar de valores pequenos para evitar uma convergência em soluções basicamente aleatórias, sendo assim TALBI (2009) elenca três pontos importantes que devem ser levados em consideração no projeto de um operador de mutação são: completude, validade e localidade. Portanto, o operador de mutação deve permitir que todas as soluções

dentro do espaço de busca possam ser alcançadas. O operador de mutação deveria produzir soluções válidas, apesar de nem sempre ser possível em casos de problemas de otimização restritivas. E, por fim, o operador deve ser responsável por uma pequena mudança (fenótipo), de forma que o tamanho da mutação seja controlado através de uma pequena mudança no genótipo.

Diferentemente de meta-heurísticas de solução única, onde seus operadores de busca são sempre unários, o operador de cruzamento é sempre binário e algumas vezes até mesmo n-ário (TALBI, 2009). O papel do operador de cruzamento é herdar algumas características de dois pais para gerar descendentes, mas a forma como isso é feito depende do tipo de representação adotada, mas de uma forma geral, existem pontos importantes para o projeto de um operador de cruzamento: a hereditariedade e a validade. O operador de cruzamento deve herdar material genético dos dois pais, de forma que um operador é um puramente recombinador se dois indivíduos idênticos geram descendentes idênticos, ou seja, possui forte herança. O operador de cruzamento deveria produzir soluções válidas, apesar de nem sempre ser possível em casos de problemas de otimização restritivas.

2.2.5. Substituição de indivíduos da população

A fase de substituição diz respeito à seleção de sobreviventes tanto dos pais quanto das populações descendentes. No algoritmo genético o tamanho da população se mantém constante a cada iteração. Tonando-se necessário substituir indivíduos de acordo com uma determinada estratégia de seleção. Porém é necessário ter em mente que, às vezes, selecionar indivíduos ruins para serem retirados da população é necessário para evitar problemas de amostragem e ter uma maior capacidade de diversificação, dessa forma Talbi (2009) apresenta duas possíveis formas de se realizar substituição de indivíduos dentro de uma população: substituição geracional e substituição em estado estacionário.

Tipo de substituição proposto por Holland (1975) em seu trabalho introdutório para os algoritmos genéticos, de forma que a substituição a partir da população de descendentes abrangerá toda a população de tamanho μ, substituindo sistematicamente toda a população.

Neste tipo de substituição, em cada geração de um algoritmo evolucionário, apenas um filho é gerado e por consequência um único indivíduo da população é substituído através de critérios adequados ao problema.

3 Problema de Pouso e Decolagem de Aeronaves

Sequenciamento e escalonamento de aterrissagem e decolagem de aviões é um problema notável e um grande desafio para o gerenciamento de tráfego aéreo. Esse problema se destaca pelo gargalo causado pela dificuldade de fazer um planejamento adequado do uso das pistas onde ocorre pouso e decolagem das aeronaves, um planejamento inadequado pode provocar atrasos nos voos, podendo gerar um efeito cascata causando prejuízos, tanto para as companhias aéreas, quanto para seus respectivos clientes (CHEN *et al.*, 2022).

No cotidiano dos aeroportos, cada aeronave possui uma janela de tempo pré-definida com horário mínimo e máximo possível de pouso e decolagem, assim como também um horário preferido ou ideal para a operação, seja ou pouso ou a decolagem. Esse tempo representa a capacidade de uma aeronave de pousar e decolar baseado em sua velocidade máxima e mínima e eficiência de consumo de combustível (IKLI *et al.*, 2021), em linhas gerais esse trabalho visa utilizar desses tempos de cada aeronave de forma a reduzir os tempos relativos aos atrasos e adiamentos nos pousos e decolagens das aeronaves.

O gerenciamento de tráfego aéreo é um componente essencial das operações de um aeroporto, e ele provê um conjunto de serviços que garantem a segurança e eficiência do fluxo aéreo (DE NEUFVILLE *et al.*, 2013). O lado operacional desse gerenciamento, que interage diretamente com a equipe técnica das aeronaves da decolagem ao seu pouso dentro do espaço aéreo controlado, é conhecido como controle de tráfego aéreo e preza pela segurança das aeronaves, garantindo conformidade com o requisito de tempo mínimo de separação e maximizar a eficiência aumentando a utilização de recursos (SOYKAN, 2016).

A Organização Internacional Civil de Aviação definiu as regras do tempo de separação, que requer que duas aeronaves, sendo atendidas consecutivamente, precisam ser separadas uma da outra por uma margem de tempo conhecido como

tempo mínimo de separação, que varia de acordo com o tipo de cada aeronave (pequena, média, grande) (SYLEJMAN ,2017).

Devido às regras de segurança exigidas pelo controle de tráfego aéreo, uma pista só pode ser utilizada por uma única aeronave por vez, sendo assim, as aeronaves que partem e chegam são sequenciadas de acordo com critérios como equidade (ordenadas por ordem de chegada) e capacidade de cada pista (FURINI *et.al.*, 2015).

Na perspectiva de um controlador local, a maneira mais fácil de agendar as operações de uma pista é através do algoritmo FCFS (do inglês, *First-Comes First-Served*), ou também conhecido como algoritmo FIFO (do inglês, *First-In First-Out*) (SOYKAN, 2016). Esta técnica de sequenciamento de aeronaves se utiliza do tempo estimado de chegada, obtido quando uma aeronave se apresenta no radar do aeroporto, e decolagem, definido pela ordem de fila das aeronaves nas posições de espera do aeroporto, associado a cada aeronave para calcular o tempo previsto de aterrissagem ou decolagem levando em consideração também os tempos mínimos de separação (IKLI *et al.*, 2021).

Contudo, apesar do FIFO ser uma estratégia eficiente em termos de implementação, normalmente ele não produz o melhor sequenciamento em termos de utilização da pista e tempo médio de atraso (CAPRÌ; IGNACCOLO, 2004). Em aeroportos de grande capacidade, uma gestão de baixo uso da pista implica em tráfegos mais congestionados e atrasos, que por consequência traz ineficiência e desperdício de recursos (SOYKAN, 2016).

3.1. Formulação do problema

Nesta seção será apresentada a formulação do problema do pouso de aeronaves originalmente proposta por Beasley *et al*. (2000).

Seja um conjunto de pistas $K = \{1, 2, ..., m\}$ e um conjunto de aeronaves $A = \{1, 2, ..., n\}$, cada aeronave $i \in A$ possui uma janela de tempo pré-definida $\left[E_i, L_i\right]$, onde $L_i > E_i$ e um tempo possível desejado de operação (pouso ou decolagem) denotado por T_i.

O problema consiste em selecionar uma pista disponível $k \in K$ e agendar um tempo de operação denotado por x_i para cada aeronave $i \in A$ sujeito a restrições operacionais, sendo assim S_{ij} é definido como o intervalo de separação requerido entre as aeronaves i e j (onde i pousa antes de j) para o pouso ou decolagem em uma mesma pista. De forma análoga, s_{ij} é utilizado para o cenário onde a aeronave i pousa antes de j e ambas estão em pistas diferentes.

Dependendo do tempo agendado para a realização da operação desejada pela aeronave serão contabilizados os custos de penalidade C_i^- e C_i^+ que representam os custos unitários para a aeronave i que pousam com antecedência e atraso em relação ao tempo alvo T_i, respectivamente.

A formulação relacionada faz o uso de variáveis binárias e contínuas, portanto trata-se de uma formulação mista.

$a_{ik} =$	$\{1\ se\ a\ aeronave\ i\ for\ pousar\ na\ pista\ k,\ 0\ caso\ contrário\}$ $i \in A, k \in K$	(2)
$Z_{ij} =$	$\{1\ se\ a\ aeronave\ i\ e\ j\ for\ pousam\ na\ mesma\ pista,\ 0\ caso\ contrário\}$ $i,\ j \in A:\ i \neq j$	(3)
$\delta_{ij} =$	$\{1\ se\ a\ aeronave\ i\ pousa\ antes\ de\ j,\ 0\ caso\ contrário\}$ $i,\ j \in A:\ i \neq j$	(4)

A partir disso, temos como função objetivo do problema a seguinte equação (5) que pretende minimizar a soma dos custos de desvios do tempo desejado T_i.

$$min\ Z = \sum_{i \in A} C_i^- x_i^- + C_i^+ x_i^+ \quad (5)$$

Sujeito a

$$E_i \leq x_i \leq L_i \qquad \forall i \in A \qquad (6)$$

$$0 \leq x_i^- \leq T_i - E_i \qquad \forall i \in A \qquad (7)$$

$$0 \leq x_i^+ \leq L_i - T_i \qquad \forall i \in A \qquad (8)$$

$$x_i^- \geq T_i - x_i \qquad \forall i \in A \qquad (9)$$

$$x_i^+ \geq x_i - T_i \qquad \forall i \in A \qquad (10)$$

$$x_i = T_i - x_i^- + x_i^+ \qquad \forall i \in A \qquad (11)$$

$$\delta_{ij} + \delta_{ji} = 1 \qquad \forall i, j \in A: i \neq j \qquad (12)$$

$$\sum_{k \in K} a_{ik} = 1 \qquad \forall i \in A \qquad (13)$$

$$Z_{ij} = Z_{ji} \qquad \forall i, j \in A: i < j \qquad (14)$$

$$Z_{ij} \geq a_{ik} + a_{jk} - 1 \qquad \forall i, j \in A: i < j, k \in K \qquad (15)$$

$$x_j \geq x_i + S_{ij} * z_{ij} + s_{ij}(1 - z_{ij}) - M\delta_{ji} \qquad \forall i, j \in A: i < j \qquad (16)$$

$$a_{ik}, z_{ij}, \delta_{ji} \in \{0, 1\} \qquad \forall i, j \in A: i \neq j, k \in K \qquad (17)$$

A restrição representada pela equação (6), representa a janela de tempo permitida para a realização da operação desejada. As restrições (7) e (8) representam o intervalo máximo e mínimo de valores que o custo de atraso ou adiantamento podem assumir, respectivamente. As restrições (9), (10) e (11) representam a relação entre as variáveis contínuas. A restrição (12) expressa a relação de precedência entre as aeronaves. A restrição (13) garante que cada aeronave realize sua operação em apenas uma única pista. A restrição (14) diz

respeito a simetria com relação às aeronaves em uma mesma pista. A restrição (15) garante que, se duas aeronaves $i\ e\ j$ realizam sua operação em uma mesma pista k então Z_{ij} é forçado a ser igual a um, da mesma forma, se duas aeronaves $i\ e\ j$ realizam sua operação em pistas diferentes então Z_{ij} é forçado a ser igual a zero. A restrição (16) garante o respeito do tempo mínimo de separação obrigatório entre as aeronaves em uma mesma pista ou em pistas diferentes.

O valor M contido na equação (16) pode ser representado pela equação (18), onde ele atua somente nos casos em que a aeronave j se antecipe e decole antes da aeronave i o tempo de decolagem de x_j será < E_i, visto que se $\delta_{ij} = 1$, logo $\delta_{ji} = 0$, isso implica que a aeronave i realizará sua operação antes da aeronave j resultando na equação (19) abaixo.

$$M = L_i + S_{ij} - T_i \qquad \forall i,\ j \in A: i \neq j \qquad (18)$$

$$x_j \geq x_i + S_{ij} * z_{ij} + s_{ij}\left(1 - z_{ij}\right) \qquad \forall i,\ j \in A: i < j \qquad (19)$$

Por fim, a restrição (17) representa os valores possíveis para as variáveis binárias definidas anteriormente.

3.2. Algoritmo Genético Aplicado ao Problema

Visando uma possível resolução do problema de sequenciamento de pouso e decolagem de aeronaves utilizando algoritmos genéticos, algumas adaptações se fazem necessárias de modo a atender os requisitos propostos.

3.2.1. Representação do indivíduo

Dado que uma solução para o problema é composta por um possível conjunto de marcações de tempo (x_i na representação feita do modelo matemático do problema) para cada aeronave que indica o tempo definido para pouso e decolagem, foi utilizada uma codificação direta de forma que o genoma de um indivíduo, o

cromossomo, é formado por uma lista de números inteiros de tamanho igual a quantidade de aeronaves envolvidas no problema, onde cada alelo guarda a informação do tempo x_i e o locuo indica a aeronave correspondente a esse tempo como pode ser observado na figura 7, em que a lista S indica um indivíduo como uma possível solução para um problema de sequenciamento de cinco aeronaves numa pista de operações, onde a aeronave 1 decolará ou pousará no tempo 115, a aeronave 2 no tempo 58, a aeronave 3 no tempo 172, a aeronave 4 no tempo 220 e por fim a aeronave 5 no tempo 248, onde o tempo é medido na unidade de tempo definida pela instância do problema.

Figura 7 – Representação do indivíduo.

$A_1 \quad A_2 \quad A_3 \quad A_4 \quad A_5$
$S = [\ 115, 58, 172, 220, 248]$

Fonte: O autor.

Foi utilizada uma codificação direta de forma que o genoma de um indivíduo representa a sua ação integralmente como fenótipo, visto que para este caso, não houve ganho, mas sim perda, computacional e da qualidade da solução adicionar a complexidade de codificação e decodificação de um genótipo para fenótipo.

3.2.2. População inicial

Visando obter uma maior variedade genética que contribui para uma maior área de busca de uma solução foi utilizada a inicialização aleatória dos indivíduos de uma forma que respeite a equação/restrição 2 do problema. Sendo assim, cada individuo é inicializado com cada alelo representado com um valor aleatório dentro do intervalo $\left[E_i, L_i\right]$.

Durante a implementação do algoritmo foi definido um parâmetro de desempenho que indica a qualidade de uma solução encontrada e representa o custo total da solução. Por se tratar de um problema de minimização, valores de desempenho elevados indicam soluções ruins que serão calculadas utilizando a

função objetivo do problema matemático através da equação (5). Como o algoritmo genético não garante que apenas soluções viáveis sejam encontradas e a necessidade de respeitar a restrição (16) foi definido um outro parâmetro chamado *undesempenho* que, de forma análoga, indica o quão viável é uma solução encontrada.

$$\sum_{i=1}\sum_{j=1} max[0,\ S_{ij}\ -\ (x_j - x_i)] \qquad \forall i,\ j \in A: i \neq j,\ x_i \leq x_j \qquad (20)$$

O parâmetro *undesempenho* é calculado utilizando a equação (20), isso implica que para uma dada solução seja dita viável seu *undesempenho* deve ser igual a 0, de forma que o quão elevado for o valor do parâmetro *undesempenho* de uma solução mais essa solução compromete a restrição (16).

3.2.3. Seleção de indivíduos

No algoritmo implementado, dois pais se unem e geram apenas um único descendente. Para escolher quais indivíduos podem se tornar um pai, foi utilizada a seleção por meio de torneios binários, onde cada torneio escolhe um dos dois pais dentre os candidatos.

Durante o torneio, da população serão selecionados dois indivíduos aleatoriamente e dentre os dois que foram selecionados aquele com o menor desempenho será o primeiro pai, caso os desempenhos sejam iguais o parâmetro de *undesempenho* será utilizado como critério de desempate selecionando o indivíduo mais viável.

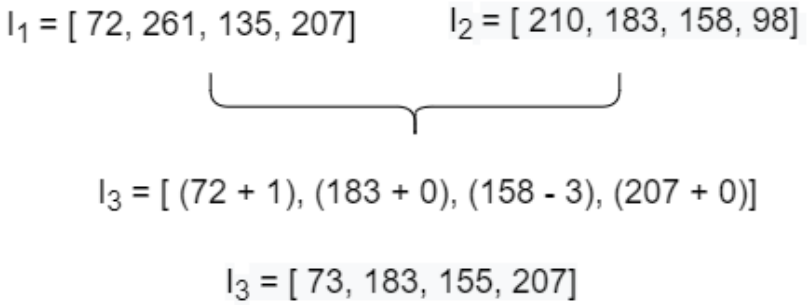
Isso garante que indivíduos melhores tenham mais chances de gerar descendentes, mas também que os indivíduos piores também tenham chances de participar do processo de reprodução.

3.2.4. Mecanismo de reprodução

Quanto ao operador de cruzamento, foi utilizado um cruzamento uniforme, onde dois pais geram um único descendente, e os valores que compõem o genoma do descendente são herdados de um ou outro pai aleatoriamente e acrescidos ou decrescidos em um valor de 0 a 3 contanto que não obstruam o intervalo possível para o tempo de pouso ou decolagem $\left[E_i, L_i\right]$, também de forma aleatória mas com uma distribuição em que o valor zero possui 50% de chance de ser escolhido e os outros 50% restantes divididos igualmente entre os valores 1, 2 e 3 com o objetivo de melhorar a solução, não depender apenas dos números aleatórios gerados na inicialização do problema e poder transitar por todas as soluções possíveis através dessa estrutura de vizinhança que pode ser observado através da Figura 8.

Neste exemplo, dois indivíduos I1 e I2 são selecionados para reproduzir e gerar descendentes. Através do operador de cruzamento, um novo indivíduo I3 é gerado que herda o material genético de seus pais sorteados aleatoriamente, que neste caso corresponde ao I1 para o primeiro alelo, I2 para o segundo, I2 novamente para o terceiro e por fim o I1 para o quarto alelo. Logo depois esses genes herdados sofrem pequenas mudanças como o acréscimo de 1 para o primeiro alelo e decréscimo de 3 para o terceiro alelo.

Figura 8 – Cruzamento de dois indivíduos.



Fonte: O autor.

Já o operador de mutação foi implementado de forma similar a seleção por roleta. Devido ao fato do algoritmo se basear, de forma geral, em números e combinações de fatores aleatórios, as soluções encontradas são bem distribuídas em questão do impacto do custo de cada alelo no indivíduo como um todo, porém alguns indivíduos apesar de terem uma boa configuração em geral apresentam algumas anomalias que prejudicam toda a sua boa configuração.

$$Mc = \frac{C_i^- x_i^- + C_i^+ x_i^+}{Z} \qquad \forall i \in A \qquad (21)$$

Sendo assim, o operador mutação atua no *locus* que há o alelo que atribui maior custo único ao indivíduo, através de uma probabilidade controlada pela equação (21), que indica que o alelo que possui o maior custo individual tem mais chances de mutar quanto mais ele influencia no custo total do indivíduo.

Caso o *locus* em questão atenda a probabilidade de mutação, será um novo alelo será inicializado randomicamente respeitando o intervalo $\left[E_i, L_i\right]$.

3.2.5. Substituição de indivíduos da população

Determinou-se o uso de um esquema de substituição populacional em estado estacionário que faz uso dos dois parâmetros de qualidade definidos nas seções anteriores (desempenho e *undesempenho*) associados a cada solução encontrada.

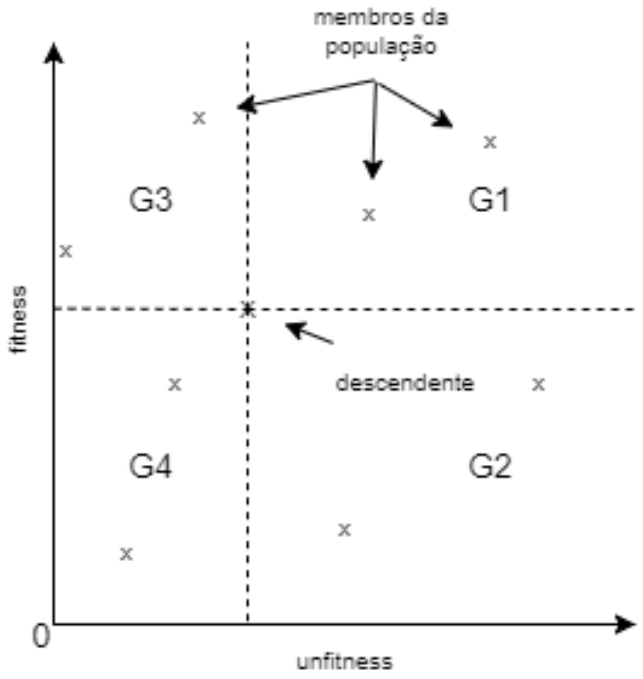
Para facilitar o entendimento, a Figura 9 mostra um novo descendente gerado e que deve ser colocado dentro da população, mas para isso deve-se decidir qual dos indivíduos que já pertencem à população deve ser substituído pela nova geração.

Considerando a Figura 9 fica claro que o novo indivíduo naturalmente dividiu a população em quatro grupos correspondentes a cada um dos quatro quadrantes criados pelo desenho.

Esses quatro grupos foram rotulados como G1, G2, G3 e G4 na Figura 9. A partir desta figura podemos inferir algumas conclusões de forma que, todos os

indivíduos do G1 são piores que o novo descendente, visto que todos possuem uma maior inviabilidade (*undesempenho*) e um maior custo (desempenho) dado pela função objetivo, todos os indivíduos do G4 são melhores que o novo membro, dado que eles possuem tanto um menor inviabilidade (*undesempenho*) quanto um menor custo (desempenho) e por fim o novo indivíduo é melhor em um quesito e pior em outro comparado aos membros do G2 e G3.

Figura 9 – Técnica de seleção para substituição de indivíduos.



Fonte: Baseado em Beasley *et al*. (2001).

Utilizando do mesmo raciocínio, quando um novo indivíduo for gerado será dividida a população dentre esses quatro grupos (G1, G2, G3 e G4) com base nos valores de referência deste novo indivíduo para estabelecer em qual 'área' da população procuramos encontrar um membro para ser substituído.

Prioritariamente são observados os indivíduos do G1 pelo fato de serem piores em ambos os quesitos comparado ao a nova geração, caso não exista membros neste grupo é levado em consideração o G2 que teriam membros com

uma inviabilidade maior, se ambos os grupos G1 e G2 estão sem nenhum candidato é a vez do G3 ser analisado por serem pior em termos de desempenho e por fim caso não haja nenhum membro em nenhum dos grupos G1, G2 e G3 resta substituir algum membro do G4.

Após a identificação do grupo apropriado, um dos membros dele será selecionado aleatoriamente para ser substituído pelo novo membro da população. Apesar de soar estranho substituir um membro do G4 por ser melhor em ambos os aspectos, torna-se necessário para manter a diversificação com fins de permitir a contínua busca por melhores soluções, ainda que a solução gerada não colabore imediatamente para este fim.

4 Experimento

Inicialmente foram realizados estudos a respeito do estado da arte dos algoritmos genéticos aplicados ao problema do ALP com foco em entender sobre a implementação e suas variações através dos trabalhos de Beasley *et al*. (2001), Sylejmani *et al.* (2017), Pasali *et al.* (2014), Caprì e Ignaccolo (2004), assim como, também, verificar outras formas de melhoria do processo de diversificação e representação do algoritmo a partir dos trabalhos de Gupta e Ghafir (2012), Lin e Hajela (1992) e Kramer (2017). Após a análise, foi implementado o algoritmo genético descrito no Capítulo 4 tendo em vista a obtenção de uma solução válida para o problema já anteriormente descrito.

Posteriormente os dados recolhidos após a realização do experimento que será feito dividido em quatro ciclos com as instâncias disponíveis serão averiguados e filtrados para que haja uma melhor compreensão da problemática. O próximo capítulo visa expor os resultados obtidos por meio de uma análise em conjunto com Pinto (2022).

4.1. Instâncias selecionadas

A execução do experimento será realizada através do uso de um conjunto de dados estáticos previamente disponíveis na OR-Library como um conjunto de dados para problemas de pesquisa operacional, descrita inicialmente por Beasley (1990). Esse conjunto de teóricos para o problema de pouso de aeronaves é formado por 13 conjuntos de instâncias listadas na Tabela 2, onde cada instância possui um número de aeronaves a ser sequenciada, como também o intervalo de tempo mínimo e máximo de cada aeronave para ter seu tempo de pouso agendado incluindo seu tempo preferido de operação T_i e seu tempo de aparecimento no radar, os custos de penalidade por unidade de tempo de atraso ou adiantamento e por fim, o tempo de separação entre cada aeronave que realizará sua operação numa mesma pista.

Tabela 2 – Conjunto de instâncias utilizadas.

Instância	Número de Aeronaves
*Airland*_1	10
*Airland*_2	15
*Airland*_3	20
*Airland*_4	20
*Airland*_5	20
*Airland*_6	60
*Airland*_7	44
*Airland*_8	50
*Airland*_9	100
*Airland*_10	150
*Airland*_11	200
*Airland*_12	250
*Airland*_13	500

Fonte: Beasley (1990).

Um exemplo de instância é mostrado através da Figura 10, onde neste caso a aeronave de índice 0 possuí um tempo para realização de sua operação mínimo de 129 e máximo de 559, acompanhado de um tempo preferido de operação dado por 155 e uma penalidade de custo por atraso e adiantamento de 10 unidades de custo por cada unidade de tempo.

Além disso, esta mesma aeronave também possui uma lista de tempos de separação que indica quanto tempo a aeronave correspondente ao índice da lista deve esperar para realizar sua operação após o pouso ou a decolagem da aeronave 0, visto que a mesma aeronave não pode pousar ou decolar mais de uma vez o tempo de separação entre a aeronave e ela mesma sempre será inalcançável neste caso igual a 99999 unidades de tempo. Logo neste caso para a aeronave de índice 1 pousar após a aeronave 0 ela deve esperar um total mínimo de 3 unidades de tempo, da mesma forma que a aeronave de índice 2 a 9 precisa esperar 15 unidades de tempo da aeronave 0.

Figura 10 – Instância *Airland*_1.

aircraft	appearance_time	earliest_landing_time	target_landing_time	latest_landing_time	penality_cost_earliest	penality_cost_latest	separation_times
0	54	129	155	559	10.00	10.00	['99999', '3', '15', '15', '15', '15', '15', '15', '15', '15']
1	120	195	258	744	10.00	10.00	['3', '99999', '15', '15', '15', '15', '15', '15', '15', '15']
2	14	89	98	510	30.00	30.00	['15', '15', '99999', '8', '8', '8', '8', '8', '8', '8']
3	21	96	106	521	30.00	30.00	['15', '15', '8', '99999', '8', '8', '8', '8', '8', '8']
4	35	110	123	555	30.00	30.00	['15', '15', '8', '8', '99999', '8', '8', '8', '8', '8']
5	45	120	135	576	30.00	30.00	['15', '15', '8', '8', '8', '99999', '8', '8', '8', '8']
6	49	124	138	577	30.00	30.00	['15', '15', '8', '8', '8', '8', '99999', '8', '8', '8']
7	51	126	140	573	30.00	30.00	['15', '15', '8', '8', '8', '8', '8', '99999', '8', '8']
8	60	135	150	591	30.00	30.00	['15', '15', '8', '8', '8', '8', '8', '8', '99999', '8']
9	85	160	180	657	30.00	30.00	['15', '15', '8', '8', '8', '8', '8', '8', '8', '99999']

Fonte: Adaptado de Beasley *et al.* (2000).

4.2. Descrição do experimento

Foram selecionadas as bases de número 1, 2, 3, 4, 5 e 8 disponibilizadas na OR-Library para a execução do experimento, que consistirá na execução de 4 ciclos, utilizando uma configuração dos parâmetros pré-definidos que serão aplicados igualmente para todas as instâncias. Os parâmetros contemplam o tamanho da população e a quantidade de iterações a serem realizadas. Os resultados encontrados foram registrados em uma planilha eletrônica. Os testes foram executados na seguinte configuração: processador AMD Ryzen 3 3200G e uma memória de 16GB RAM. O Windows 10 foi o sistema operacional utilizado para a execução do algoritmo.

Foram utilizadas como ferramentas durante o desenvolvimento do trabalho Pycharm juntamente com a linguagem de programação Python na versão 3.9, além disso, foram utilizadas as bibliotecas como numpy e Pandas. Também foi utilizado o GIT e GITHUB como ferramenta de versionamento de código, disponível em um repositório público do autor.

4.3. Resultados obtidos

A Tabela 3 mostra os resultados obtidos na execução do experimento agrupados por ciclo, utilizando a meta-heurística de algoritmo genético descrita no Capítulo 3.

Tabela 3 – Comparação dos resultados por ciclo de experimento.

Instância	N	Ciclo 1		Ciclo 2		Ciclo 3		Ciclo 4	
		Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
Airland 1	10	3080	1.10 s	1730	3.78 s	960	23.48 s	6870	1.99 s
Airland 2	15	7280	1.84 s	3120	6.02 s	2260	39.01 s	13180	3.07 s
Airland 3	20	11280	2.89 s	4490	8.58 s	1410	58.38 s	27400	4.92 s
Airland 4	20	15910	2.76 s	10460	8.27 s	6750	59.51 s	33670	5.65 s
Airland 5	20	16110	3.07 s	10990	8.30 s	6730	65.44 s	33980	5.18 s
Airland 8	50	62335	13.05 s	31290	36.97 s	17725	230.87 s	131145	27.12 s

Fonte: O autor.

A Tabela 4 mostra a disposição dos parâmetros usados para a obtenção desses resultados acima.

Tabela 4 – Parâmetros utilizados em cada ciclo do experimento.

Ciclo	Tamanho da População	Quantidade de Iterações	Operador Mutação
1	50	5000	ATIVADO
2	75	15000	ATIVADO
3	75	90000	ATIVADO
4	75	15000	DESATIVADO

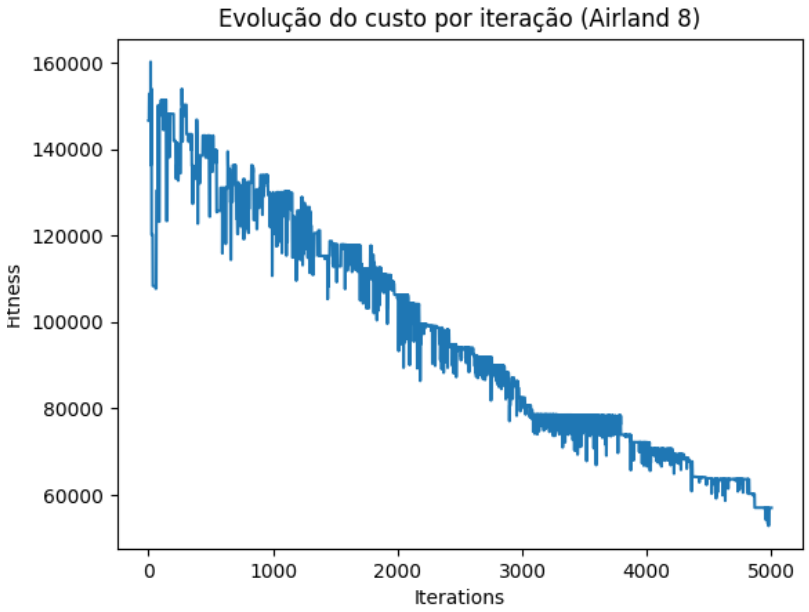
Fonte: O autor.

4.4. Análise dos resultados

O primeiro ciclo visou identificar soluções viáveis dentro de um curto espaço de busca utilizando-se de um baixo custo computacional que fica evidente pelo baixo tempo de execução comparado aos demais ciclos, visto que a pequena quantidade de membros da população utilizada e o baixo número de iterações que impediu o

algoritmo avançar em busca de soluções em regiões mais profundas do espaço de busca, especialmente para as instâncias mais complexas.

Figura 11 – Evolução do custo por iteração para a *Airland*_8 no Ciclo 1.



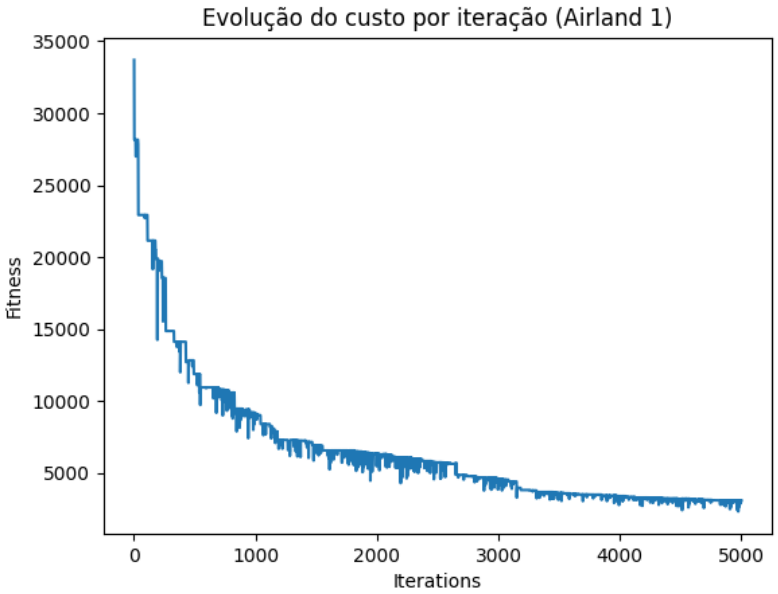
Fonte: O autor.

Isso pode ser melhor visualizado através da Figura 11 que mostra a evolução da solução de menor custo encontrada dentre os indivíduos da população ao longo das iterações para o teste realizado na instância *Airland*_8 durante o primeiro ciclo.

A Figura 12 traz a evolução do custo para o teste realizado na instância *Airland*_1. Neste primeiro ciclo apresenta um comportamento de estabilização e convergência para a melhor solução encontrada, visto ser uma instância mais simples com um espaço de busca menor.

Além disso, tendo em vista que o algoritmo implementado lida com soluções viáveis e inviáveis, fica evidente que em alguns casos mais exigentes o uso de parâmetros que restringem a evolução das soluções encontradas pode fazer com que o algoritmo não seja capaz de encontrar uma solução que atenda às restrições do problema a tempo.

Figura 12 – Evolução do custo por iteração para a *Airland*_1 no Ciclo 1.



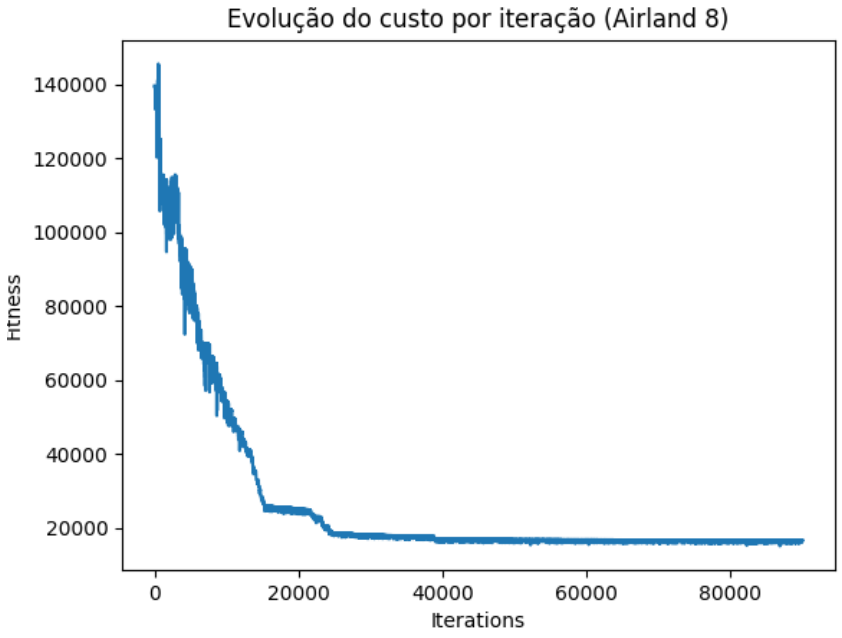
Fonte: O autor.

O segundo ciclo caracteriza-se pela tentativa de diversificar o espaço de busca aumentando o tamanho da população ao mesmo tempo que procura intensificar a busca por melhores soluções, sendo assim o segundo ciclo trouxe uma redução média de custo de cada instância de 46% comparado aos resultados encontrados no primeiro ciclo, contudo houve um aumento médio de tempo de execução de cada instância em 203%, tornando evidente o maior esforço computacional necessário para percorrer um maior espaço de busca inerente ao aumento da população e o número de iterações.

O terceiro ciclo experimenta a tentativa de super intensificar as soluções evitando um aumento da população estagnando a diversificação, devido a esse fator foram obtidos uma redução do custo médio de cada instância equivalente a 43% do custo encontrado no ciclo anterior e uma redução de 68% do custo médio se comparado ao primeiro ciclo. Além disso, foi verificado um aumento de 579% do tempo médio de execução de cada instância comparado ao tempo registrado para o

ciclo anterior, que já era esperado devido ao aumento em seis vezes do número de iterações utilizado, e um aumento de 1955% do tempo médio encontrado no primeiro ciclo.

Figura 13 – Evolução do custo por iteração para a *Airland*_8 no Ciclo 3.

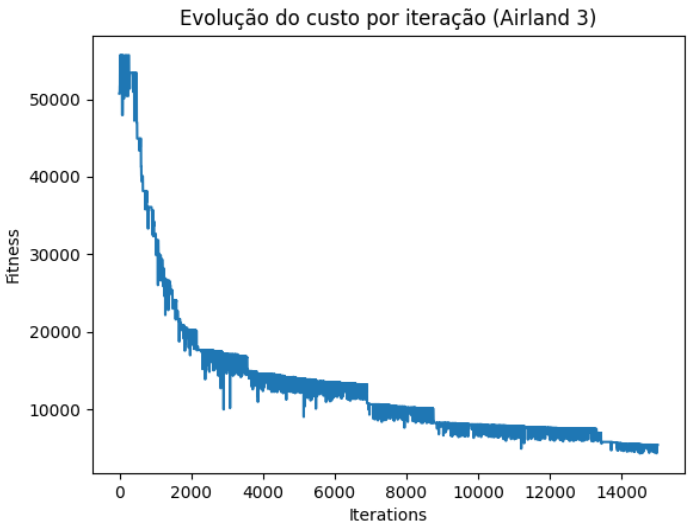


Fonte: O autor.

Os testes realizados neste terceiro ciclo revelaram a presença de um comportamento de estagnação e uma lenta progressão da solução ao longo das últimas iterações, mantendo-se próxima a melhor solução computada. Isso pode ser melhor visualizado através da Figura 13, que tornou evidente que mais da metade das iterações realizadas não contribuiu significativamente para a melhoria da solução encontrada levando apenas a um desperdício de recursos computacionais. Isso indica que um ilimitado número de iterações não faz progredir infinitamente a solução encontrada, ainda mais limitando a capacidade de diversificação com um número fixo de membros de uma população que leva a uma convergência da solução em um mínimo local mais rapidamente.

Por fim, durante o quarto e último ciclo decidiu por testar a utilidade e impacto do uso do operador mutação, observando os resultados comparado ao ciclo segundo ciclo que manteve os outros parâmetros fixos comparado a este ciclo obteve-se um aumento do custo médio total das soluções em 288% e uma redução do tempo médio de execução em apenas 39%. Portanto, apesar do operador mutação demandar mais processamento devido a necessidade de recalcular o desempenho e o *undesempenho* de todo o indivíduo sempre quando mutado, ainda sim o operador trouxe um significativo ganho relacionado ao custo das soluções encontradas.

Figura 14 – Evolução do custo por iteração para a *Airland*_3 no Ciclo 2.

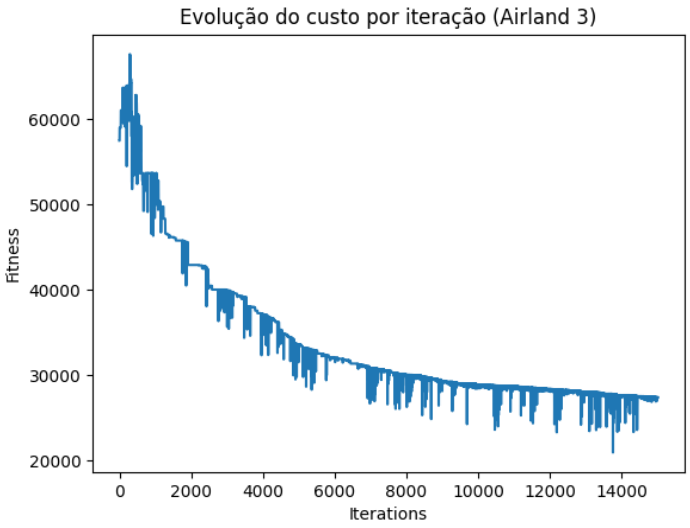


Fonte: O autor.

Comparando a evolução dos custos obtidos durante o Ciclo 2 pela Figura 14 e durante o Ciclo 4 pela Figura 15 ambos para a instância *Airland*_3, é notório que ao não se utilizar do operador mutação a curva que traz a progressão das soluções encontradas se torna mais suave, com uma lenta progressão e uma maior tendência a estabilidade e conversão para uma solução local devido à maior homogeneidade da população. É possível visualizar também que os períodos de estabilidade também ocorrem durante a execução do ciclo que se utiliza do operador, porém

esses períodos regularmente são perturbados com modificações trazidas pelo operador que da forma como foi implementado, contribui para que mais regiões sejam exploradas.

Figura 15 – Evolução do custo por iteração para a *Airland*_3 no Ciclo 4.



Fonte: O autor.

Portanto, fica claro a necessidade de conciliação entre os mecanismos de intensificação e diversificação para melhor exploração do espaço de busca visando evitar o desperdício de recursos computacionais e os ajustes e adaptações necessárias à adequação do tamanho da instância utilizada para execução do algoritmo. É necessário também deixar claro que por se tratar de um algoritmo estocástico os resultados obtidos podem variar a depender das probabilidades tomadas dentro das decisões internas do algoritmo e a primeira inicialização da população.

4.4. Análise comparativa entre ACO e GA

A Tabela 5 mostra um comparativo entre os resultados encontrados durante os três primeiros ciclos do experimento para as instâncias *a Airland*_1, *Airland*_2, *Airland*_3, *Airland*_4, *Airland*_5 e *Airland*_8 do presente trabalho utilizando

algoritmos genéticos (do inglês, *Genetic Algorithm* – GA) e os resultados encontrados pelo algoritmo implementado por Pinto (2022) para o mesmo problema utilizando a meta-heurística de colônia de formigas (do inglês, *Ant Colony Optimization* – ACO).

Tabela 5 – Comparativo de resultados entre os algoritmos ACO e GA.

Instância	N	Algoritmo	Ciclo 1		Ciclo 2		Ciclo 3	
			Custo	Tempo	Custo	Tempo	Custo	Tempo
Airland 1	10	ACO	2650	25.798 s	1750	6.974 s	1150	10.451 s
		GA	3080	1.10 s	1730	3.78 s	960	23.48 s
Airland 2	15	ACO	34210	42.665 s	29290	14.205 s	1720	33.653 s
		GA	7280	1.84 s	3120	6.02 s	2260	39.01 s
Airland 3	20	ACO	63220	1min 37 s	45210	28.720 s	1650	1min 28 s
		GA	11280	2.89 s	4490	8.58 s	1410	58.38 s
Airland 4	20	ACO	66400	1min 32 s	55610	28.680 s	4480	2min 21 s
		GA	15910	2.76 s	10460	8.27 s	6750	59.51 s
Airland 5	20	ACO	83430	1min 46 s	59680	30.483 s	4800	1min 29 s
		GA	16110	3.07 s	10990	8.30 s	6730	65.44 s
Airland 8	50	ACO	468750	20min 47 s	371995	6min 40 s	13220	53min 27
		GA	62335	13.05 s	31290	36.97 s	17725	230.87 s

Fonte: O autor.

Devido a diferenças inerentes a construção dos algoritmos, consegue-se perceber a diferença do tempo necessário para a busca pela solução principalmente se levado em consideração o segundo ciclo que traz uma melhor eficiência em termos de recursos computacionais pelo algoritmo genético, dado que o algoritmo implementado de colônia de formigas se utiliza de uma heurística construtiva permite que sua estrutura de vizinhança seja formada por regras que definem que todas as soluções encontradas no espaço de busca já são previamente válidas, comparado ao algoritmo genético que visita as soluções através de uma estrutura de vizinhança que contempla soluções viáveis e inviáveis. Sendo assim a passagem de uma solução para outra no algoritmo genético é menos custosa apesar de não garantir a viabilidade desta solução, contudo isso não é apenas um problema visto que

soluções inviáveis podem possuir genes que podem ser aproveitados de forma a contribuir para uma futura solução viável a ser formada.

A rápida progressão dos algoritmos genéticos, dado a evolução em conjunto dos membros da população, se mostraram capazes de trazer soluções mais rapidamente se comparadas a colônia de formigas que dependem da capacidade de espalhamento do feromônio ao longo das iterações para determinar um melhor caminho a ser percorrido no grafo.

Além disso, tendo em vista o último ciclo é possível visualizar o páreo entre o custo encontrado pelos algoritmos, diferenças mais expressivas são notadas em certas configurações da instância utilizada que beneficia o comportamento de determinado algoritmo, que mesmo em instâncias que possuem o mesmo número de aeronaves a serem escalonados em algumas delas se torna mais dificultoso devido ao pequeno espaçamento de tempo entre cada aeronave e a variação do custo de penalidade que torna mais punitivo um maior atraso ou adiantamento em determinadas configurações, fica evidente portanto que ambos os algoritmos, apesar de suas particularidades, são capazes de encontrar soluções viáveis para o problema em questão.

5 Conclusões e Trabalhos Futuros

O presente trabalho teve como objetivo geral de verificar a viabilidade, eficiência e aplicabilidade do uso da meta-heurística algoritmo genético para o problema escalonamento de pousos e decolagens de aeronaves.

Os resultados desse estudo apontam que é possível a aplicação do uso de metaheurísticas para a busca por uma solução viável dentro do espaço de busca, mostrando que os valores encontrados em relação ao custo da solução encontrada pela meta-heurística algoritmo genético, definida pela função objetivo dado pela formulação matemática do problema se mostraram tão bons quanto as soluções encontradas por uma outra meta-heurística populacional com heurística construtiva como é o caso da colônia de formigas em um tempo de execução não proibitivo dado a classe problemas de otimização combinatória.

Tornou-se notório também o impacto do uso operador mutação, que apesar de ser um operador que se baseia em geral na aplicação de um parâmetro que determina a sua influência na execução do algoritmo, foi implementado através de uma taxa variável que se adapta a depender das soluções encontradas até o momento que mesmo contribuindo ao aumentar o custo computacional exigido pelo algoritmo tornou-o menos suscetível a convergência prematura para um ótimo local sendo essencial nesta versão implementada para encontrar as melhores soluções disponíveis.

Quanto às dificuldades encontradas para a realização deste trabalho foram relacionadas ao entendimento inicial do problema e sua formulação matemática, a forma de adaptação e seleção de uma meta-heurística que seja capaz de atender os requisitos matemáticos, o tempo disponível para a realização do trabalho e por fim a limitação de recursos computacionais que permitisse a execução do algoritmo em instâncias maiores.

Visando os futuros desafios, propõe-se a evolução deste algoritmo de forma a alcançar soluções globais através de implementações de outros operadores que tragam uma maior capacidade de exploração do problema. Como por exemplo a construção de uma estrutura de vizinhança que visa gerar apenas soluções viáveis

diminuindo o custo computacional atual para a validação de cada novo indivíduo gerado, reinicialização da população por elitismo mantendo as melhores soluções encontradas até o momento, entre outros.

Referências

BARROS NETO, Benício de; Scarmínio, Ieda S. e Bruns, Roy E. Como fazer experimentos: pesquisa e desenvolvimento na ciência e na indústria. 2ª ed. Campinas, SP: Editora da UNICAMP, 2002.

BEASLEY, John E; KRISHNAMOORTHY, Mohan; SHARAIHA, Yazid; ABRAMSON, David. Scheduling aircraft landings—the static case. **Transportation science**, v. 34, n. 2, p. 180-197, 2000.

BEASLEY, John E. **OR-Library: distributing test problems by electronic mail. Journal of the operational research society**, v. 41, n. 11, p. 1069-1072, 1990.

BEASLEY, John E.; SONANDER, Julia; HAVELOCK, P. Scheduling aircraft landings at London Heathrow using a population heuristic. **Journal of the operational Research Society**, v. 52, n. 5, p. 483-493, 2001.

BLUM, Christian; ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, v. 35, n. 3, p. 268-308, 2003.

CAPRÌ, Salvatore; IGNACCOLO, Matteo. Genetic algorithms for solving the aircraft-sequencing problem: the introduction of departures into the dynamic model. **Journal of Air Transport Management**, v. 10, n. 5, p. 345-351, 2004.

CHEN, Zong-Gan et al. Evolutionary computation for intelligent transportation in smart cities: a survey. **IEEE Computational Intelligence Magazine**, v. 17, n. 2, p. 83-102, 2022.

COOK, Stephen A. The complexity of theorem-proving procedures. In: **Proceedings of the third annual ACM symposium on Theory of computing**. 1971. p. 151-158.

CUNHA, António, G.; TAKAHASHI, Ricardo; ANTUNES, Carlos Henggeler. **Manual de computação evolutiva e metaheurística.** Imprensa da Universidade de Coimbra/Coimbra University Press, 2012.

DARWIN, Charles. **On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life**. London: John Murray, p. 502, 1859.

DE NEUFVILLE, Richard; ODONI, Amedeo R; BELOBABA, Peter P; REYNOLDS, Tom G. **Airport systems: Planning, design, and management**. McGraw-Hill Education, 2013.

FURINI, Fabio; KIDD, Martin Philip; PERSIANI, Carlo Alfredo; TOTH, Paolo. Improved rolling horizon approaches to the aircraft sequencing problem. **Journal of Scheduling**, v. 18, n. 5, p. 435-447, 2015.

GAREY, Michael R.; JOHNSON, David S. Computers and intractability. **A Guide to the Theory of NP-Completeness**, 1979.

GUPTA, Deepti; GHAFIR, Shabina. An overview of methods maintaining diversity in genetic algorithms. **International journal of emerging technology and advanced engineering**, v. 2, n. 5, p. 56-60, 2012.

HOLLAND, John H. **Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.** MIT press, 1975.

IKLI, Sana et al. The aircraft runway scheduling problem: A survey. **Computers & Operations Research**, v. 132, p. 105336, 2021.

JONES, Antonia J. Genetic algorithms and their applications to the design of neural networks. **Neural Computing and Applications**, v. 1, n. 1, p. 32-45, 1993.

KRAMER, O. Genetic Algorithms. In: Genetic Algorithm Essentials. **Studies in Computational Intelligence**, v. 679. Springer, 2017.

LAWLER, Eugene L. Combinatorial Optimization: Networks and Matroids. **Holt, Rinehart and Winston**, 1976.

LIN, C.-Y.; HAJELA, Prabhat. Genetic algorithms in optimization problems with discrete and integer design variables. **Engineering optimization**, v. 19, n. 4, p. 309-327, 1992.

NICHOLSON, T. Optimization Techniques. In: **Optimization in Industry**, Longman Press, London. 1971.

OSMAN, Ibrahim H. An introduction to meta-heuristics. **Operational research tutorial papers**, v. 92, p. 122, 1995.

OSMAN, Ibrahim H.; LAPORTE, Gilbert. Metaheuristics: A bibliography. 1996.

PASALI, B.; ULVI, M.; BULKAN, Serol; CIL, Ibrahim. A genetic algorithm approach for aircraft landing problem, 2014.

PINTO, Matheus Phelipe Alves. **Algoritmo colônia de formigas para o problema de pouso de aeronaves em múltiplas pistas.** Orientadora: Dra. Eliane Maria Loiola. 2022. TCC (Graduação) – Curso de Engenharia de Computação, Universidade de Pernambuco, 2022.

SILVER, Edward A.; VICTOR, R.; VIDAL, V.; DE WERRA, Dominique. A tutorial on heuristic methods. **European Journal of Operational Research**, v. 5, n. 3, p. 153-162, 1980.

SOYKAN, Bulent. **A hybrid Tabu/Scatter Search algorithm for simulation-based optimization of multi-objective runway operations scheduling.** Old Dominion University, 2016.

SYLEJMANI, Kadri; BYTYÇI, Eliot; DIKA, Agni. Solving aircraft sequencing problem by using genetic algorithms. **Intelligent Decision Technologies**, v. 11, n. 4, p. 451-463, 2017.

TALBI, El-Ghazali. **Metaheuristics: from design to implementation**. John Wiley & Sons, 2009.

WOLPERT, David H.; MACREADY, William G. No free lunch theorems for optimization. **IEEE transactions on evolutionary computation**, v. 1, n. 1, p. 67-82, 1997.