Algoritmos genéticos aplicados na estruturação de rotas de manutenção no setor de geração de energia elétrica

Gabriel Salles do Amaral - 52496

Trabalho realizado sob a orientação de

Profa . Dra Ana Isabel Pereira

Profa . Dra Marjorie Maria Bellinello

Mestrado em Engenharia Industrial - Engenharia Mecânica

2022-2023

Algoritmos genéticos aplicados na estruturação de rotas de manutenção no setor de geração de energia elétrica

Relatório da tese de

Mestrado em Engenharia Industrial - Engenharia Mecânica

Escola Superior de Tecnologia e Gestão

Universidade Tecnológica Federal - Paraná

Gabriel Salles do Amaral - 52496

2022-2023

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Declaro que o trabalho descrito neste relatório é da minha autoria e é da minha vontade que o mesmo seja submetido a avaliação.

Gabriel Salles do Amaral - 52496

Agradecimentos

Agradeço a todos, os que de alguma forma, tornaram possível a realização deste trabalho.

À minha mãe Vírginia, por ser um exemplo para mim, pelos incomparáveis esforços e por sempre fazer tudo o que lhe estava ao alcance para me proporcionar as melhores oportunidades. Agradeço também a minha família por todo o apoio, carinho e incentivo para minhas aventuras.

Agradeço a alguns professores que tive a honra de conhecer ao longo de minha trajetória acadêmica, em especial, as professoras Ana Isabel e Marjorie Bellinello, orientadoras desta dissertação, pelos esclarecimentos, paciência e apoio constante no desenvolvimento deste trabalho. Ao professor David Nuñez pelos conselhos, palavras encorajadoras e todo aprendizado ao logo destes anos.

A todos os amigos que fazem parte da minha jornada, por poder ter seu apoio, carinho e amizade nos momentos de alegria e principalmente nos de dificuldade, por contribuírem para a formação da pessoa que sou hoje.

Agradeço a UTFPR, pela honra de ter feito parte dessa grande instituição, que me proporcionou crescimento pessoal e profissional. Em especial pelos projetos que tive a oportunidade de participar, em que tive vivências extraordinárias e conheci pessoas incríveis que sempre me incentivaram a buscar o melhor.

Meus agradecimentos ao IPB, pela estrutura, vivência internacional e oportunidade de fazer parte desta instituição.

Àqueles que injustamente me esqueci de referir.

A todos um muito obrigado.

Resumo

As centrais hidroelétricas exercem um papel fundamental na geração de energia elétrica brasileira, sendo indispensável a estruturação de uma política de manutenção assertiva, de forma a aumentar a confiabilidade e disponibilidade dos ativos industriais pertencentes a elas. Dentro desse contexto um dos desafios da manutenção é o planeamento de rotas de manutenção, uma vez que um sistema hidroelétrico possui milhares de equipamentos, cada um com suas especificidades. Este trabalho tem como objetivo desenvolver uma ferramenta para estruturação de rotas de manutenção inteligentes utilizando conceitos da tecnologia 4.0, mais especificamente algoritmos genéticos, para uma tomada de decisão assertiva em relação as estratégias de manutenção.

Palavras-chave: Algoritmos genéticos, Indústria 4.0, Políticas de manutenção, Rotas de manutenção.

Abstract

Hydroelectric power plants have a fundamental role in the generation of Brazilian electricity, and it is essential to structure an assertive maintenance policy in order to increase the reliability and availability of industrial assets belonging to them. Within this context, one of the maintenance challenges is the prioritization of critical components, since a hydroelectric plant has thousands of pieces of equipment. This research work aims to develop a critical component prioritization tool using 4.0 technology concepts, more specifically genetic algorithms, for an assertive decision-making in relation to maintenance strategies.

Keywords: Genetic algorithms, Industry 4.0, Maintenance policies, Prioritization of critical components.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de abreviaturas

AG Algoritmo Genético. xii, xiv, 11, 12, 14–16, 29, 30, 32, 33, 47–53, 61, 65

AHP Analytic Hierarchy Process. 8

ANEEL Agência Nacional de Energia Elétrica. 18

DNA Deoxyribonucleic Acid. 11

ERSE Entidade Reguladora dos Serviços Energéticos. 18

FMECA Failure Modes and Effects Criticality Analysis. 8

FMSA Failure Mode Symptoms Analysis. 8

HAZOP Hazard and Operability Study. 8

IA Inteligência Artificial. 10, 11, 23

IOT Internet of Things. 10

MTBF Tempo Médio entre Falhas. 8

MTTR Tempo Médio para Reparo. 8

MW Megawatt. 37

NumPy Numerical Python. xiv, 25–28

OEE Eficiência Global do Equipamento. 8

Pandas Python Data Analysis Library. xiv, 25–27

PyPi Python Package Index. 25

RBM Restricted Boltzmann machines. 28

RCM Reliability Centered Maintenance. 7

RUL Remaining Useful Lifetime. 6

SciPy Scientific Python. xiv, 25, 26, 28

TPM Total Productive Maintenance. 7, 8

UG Unidade Geradora. 2, 17

UNVPM Universitá Politecnica Delle Marche. 38

USP Universidade de São Paulo. 38

Capítulo 1

Introdução

1.1 Contextualização do problema

O cenário económico atual e a competitividade global obriga as empresas a investir mais nos recursos do fortalecimento de seus processos produtivos e sistemas de apoio para manter a estabilidade e criar vantagens competitivas nas organizações. Neste cenário, a gestão da manutenção dos ativos empregados nos processos produtivos torna-se um dos principais motores para melhorar os resultados das organizações, desempenhando um papel importante no suporte às estratégias de negócios e de operação [72].

A gestão da manutenção nos sistemas de produção modernos não se trata apenas da restauração dos ativos físicos ao seu estado operacional após uma falha. Como processo de apoio ao negócio, engloba o planeamento, organização, implementação e controle de todas as atividades de manutenção, visando a aplicação de ações preventivas evitando a ocorrência de falhas, aumentando assim o tempo de vida útil dos ativos.

No Brasil, um dos setores que desempenham um papel essencial para economia e manutenção da vida moderna são as Centrais Hidroelétricas. Este tipo de energia representa mais de 60% da matriz energética brasileira [19].

Um complexo físico para geração de energia elétrica contempla ativos mecânicos, elétricos e eletrónicos agregados a uma estrutura civil complexa. Estes ativos sofrem condições

variadas de operação, além de um ambiente extremamente agressivo para os materiais de construção, o que pode ocasionar falhas e interromper a geração de energia [41, 66].

Uma política de manutenção assertiva é necessária não apenas para garantir a integridade e funcionamento correto desses equipamentos, mas também para conferir uma alta confiabilidade e disponibilidade dos mesmos [76].

Assim, é necessário estruturar uma política de manutenção adequada aos ativos industriais usados no setor hidroelétrico. Na modelação do processo de decisão na era da indústria 4.0, existem técnicas que podem ser exploradas como *machine learning*, rede neural, métodos multiobjetivos, algoritmo genético, rede bayesiana, entre outras.

A agregação dessas ferramentas no processo decisório industrial possibilita otimização dos processos decisórios na manutenção industrial, visando a tomada de decisão robusta e eficaz [11].

Este trabalho visa estruturar uma ferramenta que identifica rotas de manutenções assertivas dos equipamentos que englobam uma Unidade Geradora (UG) do tipo Kaplan por meio da aplicação da modelação decisória com algoritmo genético. Esta priorização de rotas e, consequentemente, de componentes direciona a uma estrutura adequada de política de manutenção adequada, visando conferir confiabilidade no sistema de geração de energia elétrica com segurança pessoal para os funcionários e meio ambiente.

1.2 Objetivos

Desenvolver um algoritmo, fundamentado com tecnologia 4.0 para selecionar atividades de manutenção e otimizar as rotas de manutenção dos equipamentos de um hidrogerador Kaplan, visando estruturar uma política de manutenção adequada que garanta confiabilidade do sistema. Além disso, os objetivos específicos estão listados abaixo:

- Efetuar a revisão da literatura acerca do tema do trabalho;
- Abordar sobre os equipamentos que compõem o sistema de geração de energia;

- Desenvolver a modelação do processo decisório associado ao sistema de geração de energia;
- Aplicar um algoritmo populacional, por exemplo o algoritmo genético para a resolução do problema definido. Este processo determina as rotas de manutenção dos componentes críticos que compõem a planta hidroelétrica;
- Aplicação linguagem de programação Python para estruturar o código de priorização das atividades a serem realizados manutenção;
- Explorar outros algoritmos para resolução do problema;
- Conclusão e sugestões para trabalhos futuros.

1.3 Estrutura do Documento

Este trabalho está organizado em seis capítulos e dois apêndices. No segundo capítulo apresenta-se, de forma detalhada, o referencial teórico utilizado no desenvolvimento do trabalho que engloba conceitos de manutenção, algoritmos genéticos e estrutura de uma unidade hidrogeradora.

No terceiro capítulo é abordado sobre a linguagem de programação usada no estudo, assim como, as suas principais bibliotecas e os problemas de otimização que inspiraram no desenvolvimento do programa implementado na pesquisa.

Já no quarto capítulo apresenta o desenvolvimento do estudo de caso, mais especificamente o desenvolvimento dos algoritmos genéticos e suas aplicações.

No quinto capítulo apresenta a análise dos resultados e sua discussão.

Por fim, no sexto capítulo descreve as conclusões e recomendações para trabalhos futuros.

Capítulo 2

Revisão da literatura

Este capítulo descreve alguns conceitos técnico-científicos aplicados no desenvolvimento desse trabalho, que compreendem: conceitos de manutenção e sua classificação (principais tipos), métodos para desenvolvimento de políticas de manutenção para ativos industriais; tecnologias da indústria 4.0 na gestão industrial, nomeadamente alguns algoritmos populacionais e estrutura de unidades hidrogeradoras de energia elétrica.

2.1 Conceitos de manutenção industrial

A definição de manutenção é um conceito bem estabelecido na literatura. A norma portuguesa EN 13306:2007 define manutenção como a "combinação de todas as ações técnicas, administrativas e de gestão, durante o ciclo de vida de um bem, destinadas a mantê-lo ou repô-lo num estado em que ele pode desempenhar a função requerida." Assim, a manutenção tem como objetivo garantir o funcionamento dos equipamentos presentes nas instalações industriais de forma a realizarem a função para a qual foram projetados [18].

Existem três tipos principais de manutenção, a manutenção corretiva é aquela que visa recolocar um equipamento, componente ou instalação em operação após a ocorrência de uma falha a fim de executar a sua função requerida. Pode ser classificada ainda em manutenção corretiva planeada e não planeada, isto é, se foi previamente programada ou

não [78, 18].

Já a manutenção preventiva caracteriza-se pela conservação da maquinaria por meio de planos de manutenção periódicos, possibilitando a redução de custos de reparo e substituição de peças com a melhoria gerada na disponibilidade e confiabilidade do sistema produtivo. Esta estratégia reduz a necessidade da manutenção corretiva, provocando um aumento de vida útil dos ativos industriais [15, 24].

E por fim, a manutenção preditiva que procura prever quando um equipamento ou componente irá falhar para então realizar uma atividade de manutenção de forma a otimizar o tempo de vida do equipamento. Este tipo de manutenção fundamenta-se na monitorização de parâmetros como: temperatura, fadiga, vibração, ruído, composição química etc., por meio de instrumentos, de forma a acompanhar o funcionamento de cada mecanismo periodicamente [55, 24].

Dessa forma, através de indicadores, como o tempo de vida útil restante, *Remaining Useful Lifetime (RUL)*, as empresas conseguem estimar e prever quando um determinado componente apresentará falha, programando as intervenções de manutenção com maior assertividade, tornando possível aproveitar o máximo de operação de cada ativo com mínimo de trocas de componentes, o que, consequentemente, reduz os custos do setor [55, 73].

2.2 Métodos para estruturar política de manutenção de ativos industriais

A manutenção é definida como todas as ações necessárias para reter um item ou ativo, ou restaurá-lo, em uma condição na qual ele atinge seu potencial de serviço originalmente especificado. A gestão da manutenção é um fator chave nas centrais hidroelétricas, por isso deve estar integrada e alinhada com os demais departamentos para conservar o valor do ativo e garantir os serviços.

A disponibilidade e confiabilidade dos sistemas de geração de energia elétrica podem

ser mantidas por meio de políticas de manutenção adequadas, permitindo antecipar falhas e eliminar suas causas. As atividades de manutenção são o processo mais crítico a ser considerado nessas centrais hidroelétricas. Essas atividades causam interrupções de geração planeadas ou não planeadas [13].

As políticas de manutenção surgem com o viés de estender a vida útil de equipamentos, dessa forma reduz custos relacionados a reparos ou trocas prematuras de ativos, sendo algo atrativo para as indústrias. Portanto, a inexistência de uma política de manutenção robusta não apenas reduz a eficiência do sistema, mas também aumenta a probabilidade de falhas do mesmo [15].

A gestão da manutenção de ativos industriais requer a adoção de decisões assertivas quanto ao método de tomada de decisão para determinar ativos prioritários para monitorização preventiva e/ou preditiva, visando desenvolver uma política de manutenção eficaz que garanta elevados níveis de produtividade, otimizando custos e recursos de manutenção [17].

[43] detalha a literatura que aborda essa temática, classificando a gestão da manutenção em sub áreas, em que algumas metodologias se destacam como, por exemplo, a Manutenção Centrada na Confiabilidade - *Reliability Centered Maintenance (RCM)* e Manutenção Produtiva Total - *Total Productive Maintenance (TPM).*

O RCM é focado em garantir a confiabilidade dos ativos industriais, isto é, manter os equipamentos ou máquinas funcionando dentro dos parâmetros em que foram projetados ao longo das suas respectivas vidas úteis. Para alcançar esse objetivo esse método leva em consideração alguns fatores como: consequências, probabilidade, histórico e severidade das falhas. Portanto, a estratégia de manutenção é estruturada de acordo com o modo de falha, em outras palavras, a criticidade e, consequentemente, a priorização dos equipamentos é baseada em seus respectivos modos de falhas, com cada um deles possuindo suas atividades de manutenção específicas, o que torna esse método assertivo e melhora a confiabilidade dos ativos industriais como um todo [70].

[45] aplicou o RCM com o objetivo de otimizar a gestão da manutenção do sistema de

uma turbina de uma Central Hidroelétrica, através do *Failure Modes and Effects Criticality Analysis (FMECA)* foi possível identificar os subsistemas que apresentaram maior índice de incidentes, definir as respectivas ações necessárias de manutenção, permitindo um melhor controle por parte da manutenção, assim como, reduzir os seus respectivos custos.

O TPM, por sua vez, surgiu no Japão alinhado a metodologia *lean manufacturing*, procurando que a manutenção seja produtiva, sem perdas, falhas e gastos desnecessários. É uma metodologia que otimiza os modelos tradicionais gestão através da procura contínua por eliminar desperdícios e melhorar os colaboradores, processos e qualidade dos serviços e/ou produtos oferecidos [25].

[52] aplicou o TPM em uma indústria de alimentos. Por meio dos indicadores utilizados na metodologia, como Eficiência Global do Equipamento (OEE), Tempo Médio entre Falhas (MTBF) e Tempo Médio para Reparo (MTTR) foi verificado que as maiores perdas da empresa ocorriam devido à quebra das máquinas, então foi estruturado atividades de manutenção mais assertivas, seguindo os conceitos do TPM e após a implementação as taxas de avarias, refugos, retrabalho diminuíram e a eficiência das máquinas aumentaram, indicando o sucesso da metodologia.

[29] desenvolveu um *smart system* apoiado em algoritmos de aprendizagem (*machine learning*) capaz de avaliar as perdas que ocorrem no processo industrial por ocorrência de falhas (manutenção corretiva) e operacional e, assim, definir para o momento mais oportuno de para intervenção manutenção planeada em grupos específicos de ativos críticos.

Uma política de manutenção pode ser composta de várias metodologias e ferramentas, [50] demonstra isso, desenvolvendo um método para identificar os componentes críticos de uma central hidroelétrica com hidrogerador Kaplan, localizada no norte do Brasil. Para isso foi utilizado diversas ferramentas como elaboração da árvore funcional da central hidroelétrica em questão, *Hazard and Operability Study (HAZOP)*, *Failure Mode Symptoms Analysis (FMSA)* que é uma variação do FMEA e FMECA e o *Analytic Hierarchy Process (AHP)* como método decisório para ordenar os componentes mais críticos e, a partir disso, desenvolver uma política de manutenção, demonstrando como as estratégias

de manutenção são versáteis e podem se complementar, visando o objetivo principal da manutenção, que é aumentar a disponibilidade dos ativos industriais.

[14] propôs uma estratégia de manutenção preditiva em que o monitorização e previsão de falhas das turbinas eólicas ocorrem por meio do algoritmo *Random Forest*, que é um método de classificação através da construção de árvores de decisão, também foi utilizado de *Big Data Analysis* para análise dos dados.

As ferramentas da indústria 4.0 não se limitam apenas a previsão de falhas e priorização de componentes críticos, como demonstrado em seu trabalho, [39] desenvolveu um modelo capaz de prever o custo relaciona a manutenção de um edifício com a utilização de algoritmos genéticos. O que reforça que as ferramentas da indústria 4.0 podem alavancar os resultados associados a manutenção de várias formas.

[63] realizou uma revisão da literatura a respeito das ferramentas e metodologias aplicadas para tomada de decisão na manutenção. Foi identificado que existe uma procura para desenvolvimento e aprimoramento de tecnologias voltadas a manutenção preditiva, em que exista um monitorização das condições de equipamentos e máquinas em tempo real. No seu estudo foi demonstrado que os algoritmos genéticos ainda são pouco explorados para estruturação de políticas de manutenção, destacando que é uma ferramenta capaz de otimizar soluções voltadas para tomada de decisão.

2.3 A importância do planeamento de rotas de manutenção

Uma das etapas cruciais da gestão da manutenção é a programação das atividades que serão executadas pela equipe de manutenção, sejam elas vinculadas a manutenção corretiva, preventiva ou preditiva. Independentemente da modalidade da manutenção, é essencial que sejam planeadas de forma eficaz, procurando atingir os resultados e objetivos do setor, além de impactar positivamente nos indicadores e performance da indústria [5, 44, 47].

Em instalações complexas como centrais hidroelétricas é vital que sejam seguidos os

cronogramas de manutenção, de forma a alcançar sua função principal, que é a geração de energia. Por possuir uma grande quantidade de equipamentos, cada um com suas características e especificidades, é essencial que exista um planeamento assertivo das atividades da manutenção visando garantir a confiabilidade do sistema [53].

Durante o planeamento das atividades, é importante que seja levado em consideração a disposição do maquinário da indústria, uma vez que pode afetar o desempenho da equipa de manutenção, já que existem inúmeras possibilidade diferentes de rotas a serem seguidas para efetuar as atividades de manutenção. Portanto, é relevante determinar rotas otimizadas de manutenção, melhorando aspectos como o tempo de deslocamento, aumentando o número de máquinas a serem trabalhadas e proporcionando segurança a equipa executora, o que eleva o rendimento da indústria como um todo [47, 51].

Um exemplo da efetividade de otimização de rotas de manutenção é o trabalho de [1] que propõem otimização de rotas de manutenção para parques eólicos *offshore*, que são grandes instalações em alto mar, a fim de minimizar custos e manter uma alta disponibilidade das turbinas eólicas, através do *Ant Colony System*, que é um algoritmo inspirado no comportamento das formigas em procura de um caminho entre sua colónia e fontes de alimento.

2.4 Algoritmo genético aplicado na tomada de decisão industrial

A sociedade como um todo, vem avançando e inovando em termos de tecnologia, fenómeno conhecido como quarta revolução industrial, comummente chamada de Indústria 4.0. A filosofia da quarta revolução industrial visa a integração e automação da tecnologia à indústria, através de ferramentas como *Internet of Things (IOT)*, *Big Data Analytics* e Inteligência Artificial (IA) que conectam o mundo físico ao online, buscando tornar uma indústria inteligente [79].

[49] ressaltam a importância de traçar estratégias voltadas a manutenção que aproveitem essas tecnologias já existente, visto que o custo da manutenção pode chegar a 25% do custo operacional de uma indústria.

A integração exigida pela indústria 4.0 procura proporcionar avanços no monitorização das condições dos equipamentos, serviços e reparos remotos, determinar o tempo de vida útil restante dos componentes e desenvolver cada vez mais a manutenção preditiva [42].

Em especial, a IA tem ganho destaque dentro da área de manutenção como uma ferramenta poderosa para resolução de problemas. Sua aplicação na gestão e planeamento possibilita desenvolver uma manutenção inteligente. A IA pode ser entendida como um ramo da ciência da computação que tem como objetivo desenvolver programas que permitam que máquinas executem atividades que necessitem de inteligência humana, para alcançar esse propósito existem várias técnicas e, uma delas, é o AG [23].

O AG é um método de procura baseada nos princípios de seleção natural e evolução biológica, com uma relativa simplicidade de implementação e eficácia em realizar procuras globais em ambientes adversos, sendo desenvolvida em meados de 1970 por J. Holland [31].

Da perspectiva biológica, a capacidade de um organismo sobreviver em um ambiente é determinada pelo seu *Deoxyribonucleic Acid (DNA)*, que é uma combinação do DNA de seus pais, herdando algumas de suas características e outras que surgem devido a recombinação desses DNA's. O conjunto desses traços podem contribuir para o aumento da probabilidade de adaptação dessa prole e, consequentemente, sobrevivência. Tais características serão passadas para a próxima geração e assim sucessivamente, melhorando, com o passar do tempo, a capacidade de sobrevivência dessa espécie [4].

Por se basear nos processos evolutivos das espécies, existem alguns termos correspondentes entre a linguagem adotada nos algoritmos genéticos e na biologia, mostrados na Tabela 2.1, abaixo:

Além disso, outro termo utilizado são os operadores genéticos. Tem como objetivo alterar características dos indivíduos das populações, seja para diversificar as futuras gerações ou manter características obtidas de gerações passadas. Existem três tipos de operadores genéticos: seleção, recombinação e mutação, que serão explicadas nos próximos

Tabela 2.1: Correspondência entre vocabulário biológico e computacional. Adaptado de [35].

Termo biológico	**Termo computacional**
Cromossoma	Indivíduo
Gene	Característica
Alelo	Valor
Lócus	Posição
Genótipo	Estrutura
Fenótipo	Conjunto de parâmetros

parágrafos [31].

Os AG's trabalham com uma população de indivíduos, que representam uma possível solução para o problema em questão, em que os parâmetros são representados por um conjunto de símbolos. É atribuído uma pontuação para cada indivíduo, de acordo com a qualidade que essa solução representa, os indivíduos mais preparados serão reproduzidos com outros indivíduos, gerando uma nova geração, ou seja, a cada iteração é esperado que uma a qualidade dessa nova população aumente, levando a uma solução ótima [35].

As etapas de execução de um AG para tomada decisão estão ilustradas na Figura 2.1 abaixo:

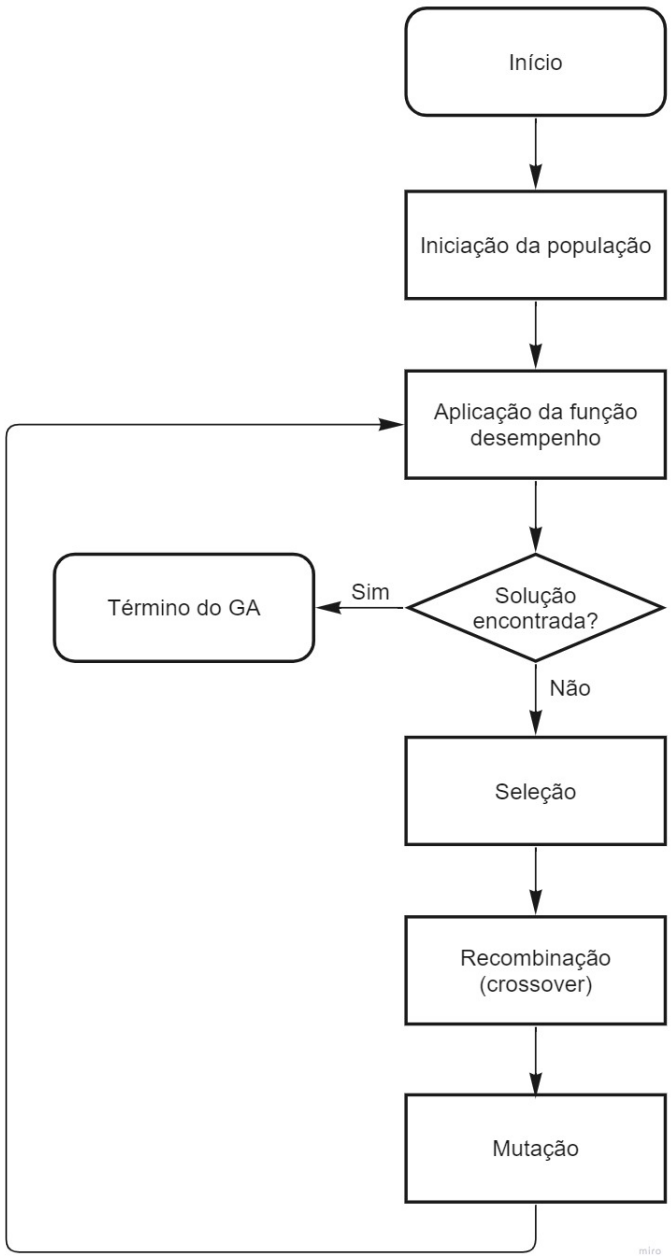


Figura 2.1: Processo de um algoritmo genético. Adaptado de [71].

I. **Iniciação da população:** nessa etapa é gerada a população inicial, sendo definido seu tamanho, o número de iterações, tipos de cromossomas e os operadores genéticos [71].

II. **Aplicação da função desempenho (*fitness*):** tem como objetivo determinar o

quão boa uma solução é, avaliando cada indivíduo de uma população por meio de um valor específico. A probabilidade de um indivíduo se propagar para a próxima geração é proporcional ao seu valor *fitness*, que após cada interação espera-se obter indivíduos cada vez melhores [35].

[31] ressaltam que a função *fitness* deve ser definida com atenção, deve-se conhecer muito bem o problema a ser resolvido, isto é, eventuais restrições e objetivos a serem alcançados, para que o AG seja capaz de fornecer uma solução adequada.

III. **Seleção:** nessa etapa são selecionados os indivíduos geradores da próxima geração, em que suas possibilidades de serem escolhidos aumentam de acordo com o seu valor *fitness* [35].

IV. **Cruzamento (*crossover*):** nesse processo que ocorre a recombinação aleatória das características (cromossomas) dos pais para formar um novo indivíduo. O objetivo do cruzamento é a troca de informações entre diferentes soluções candidatas [4]. A Figura 2.2 abaixo ilustra o rearranjo dos genes:

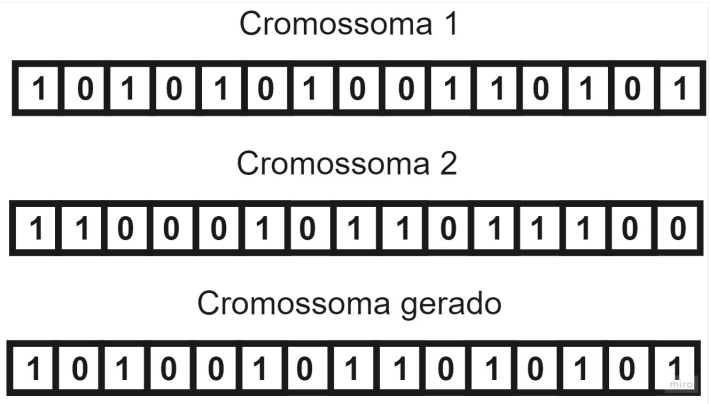


Figura 2.2: Cruzamento (*crossover*) entre dois cromossomas. Adaptado de [23].

V. **Mutação:** ocorre com o intuito de aumentar a diversidade da nova população gerada. Envolve a seleção aleatória de genes dentro dos cromossomas, atribuindo valores dentro do intervalo selecionado. Esta etapa permite a entrada de novos indivíduos na população restringindo uma convergência prematura ou restaurando

características desejadas que possam ter sido removidas da população muito cedo [31]. A Figura 2.3 apresenta um exemplo de como a mutação pode ocorrer.

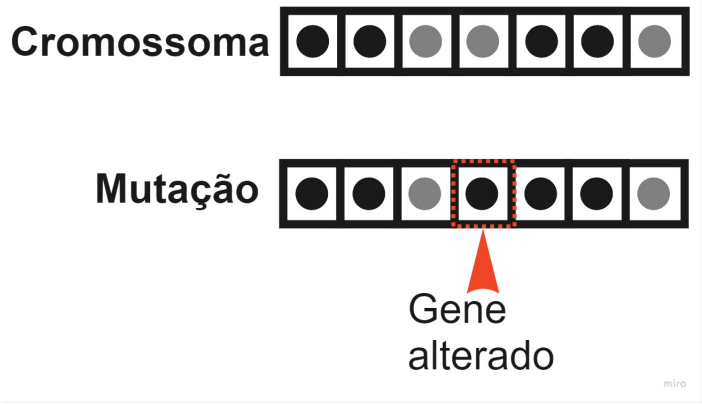


Figura 2.3: Mutação de gene. Adaptado de [4].

VI. **Término do AG:** um AG tem como objetivo encontrar uma solução ótima ou próximo disso, portanto, essa etapa é a finalização do algoritmo genético quando essa meta é alcançada.

[71] propôs uma abordagem utilizando dados obtidos dos sinais de correntes dos motores para identificar falhas de rolamentos em motores de indução. Foi aplicado o AG para selecionar os dados mais importantes da base de dados e, posteriormente, usado *machine learning* para classificar e quantificar as falhas de rolamentos, obtendo uma eficácia de 97%.

[33] procurou otimizar o alinhamento ferroviário vertical, com o objetivo de minimizar os custos tanto para os utilizadores quanto para operação e construção, através de algoritmos genéticos e modelos de simulação. Como aponta o autor, projetos de alinhamento vertical são importantes pois afetam decisões operacionais, como, por exemplo, tempo de viagem e o consumo de energia de tração e frenagem.

[77] propôs dois métodos utilizando AG's para selecionar o caminho com menor tempo para realizar a manutenção de redes de energia em cidades localizadas na China, considerando, por exemplo, nível de gravidade e atribuindo vários trabalhadores. Ambos métodos

apresentaram um desempenho melhor que as abordagem tradicionais para traçar as rotas de manutenção.

Os AG's são um método poderoso de procura podendo ser aplicados para diversos problemas de otimização, por terem uma implementação relativamente simples e flexibilidade se tornam atrativos para solucionar problemas [35]. Por ter essas características tem um potencial de aplicação elevado para solução de problemas na manutenção.

2.5 Estrutura e funcionamento de Unidades Hidrogeradoras de Energia Elétrica

Uma Central Hidroelétrica converte a energia potencial da água em energia mecânica pela turbina que, posteriormente, movimenta o rotor de um gerador elétrico para produzir energia elétrica. Esse processo, brevemente explicado, envolve diversos componentes, a Figura 2.4 abaixo ilustra a estrutura de uma central hidroelétrica [12].

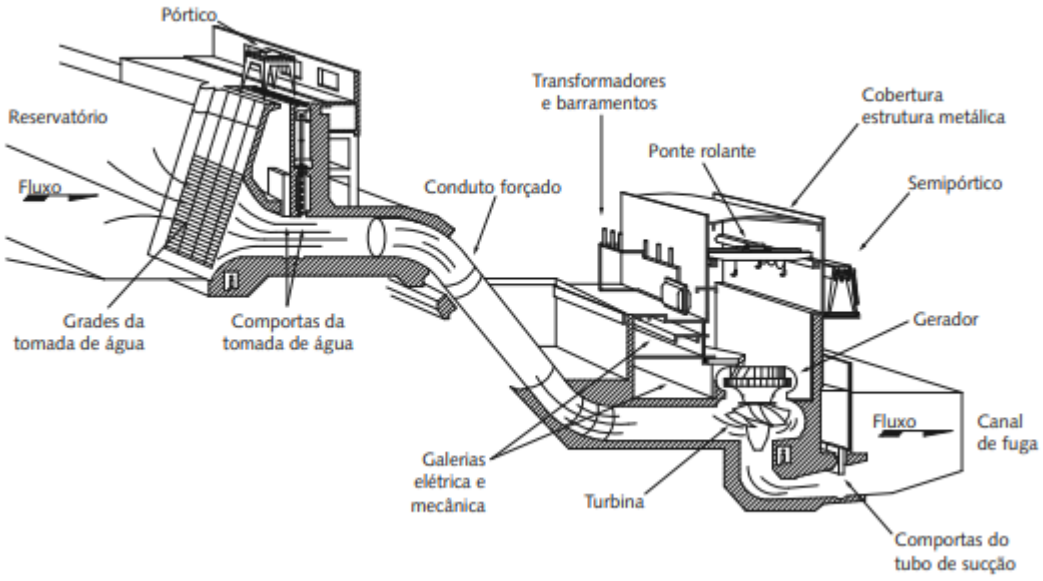


Figura 2.4: Ilustração de uma Central Hidroelétrica. [56].

As principais partes de uma Central Hidroelétrica são a barragem, a tomada de água,

o vertedouro, as comportas, a casa de força e as turbinas [60].

A Barragem é uma estrutura que tem como finalidade represar e armazenar a água de um rio. Ela também tem como objetivo proporcionar o desnível necessário para que as turbinas das centrais possam movimentar-se. Por sua vez, o reservatório é uma infra-estrutura capaz de controlar o nível de água de forma aumentá-lo para aproveitamento elétrico ou evitar inundações [58].

[64] destacam que as tomadas de água são estruturas que tem a finalidade de captar e conduzir a água armazenada nas albufeiras aos canais de adução e, por fim, para as turbinas. Além disso, são responsáveis por evitar a entrada de corpos flutuantes que possam danificar as turbinas e também, quando necessário, fechar essa entrada de água, atuando como um mecanismo de segurança.

O vertedouro tem como objetivo reduzir o nível de água dos reservatórios, evitando que o nível máximo de água suportado por eles seja ultrapassado em épocas de cheias ou enchentes, além de ser responsável por descarregar a água não usada para geração [56, 69].

Já as comportas são os componentes que isolam a água do sistema final de produção de energia elétrica, o que torna possível atividades e operações voltadas a manutenção e operação [60].

A casa de força é o local em que abriga as máquinas e equipamentos eletromecânicos, de forma a possibilitar ações como montagens, desmontagens, operação e manutenção [34].

E as Turbinas, ou UG's representam o principal ativo estruturante do processo de geração de energia. As turbinas tem como objetivo principal transformar energia hidráulica em energia mecânica. Elas podem ser classificas em relação a variação da pressão estática em ação ou impulso e reação [36].

Quando o escoamento de água através do rotor ocorre sem variação de pressão é classificada como turbina do tipo ação, por exemplo, a turbina Pelton. Por outro lado, se esse escoamento ocorre com variação de pressão é do tipo reação, como as turbinas Kaplan e Francis [36].

2.5.1 Hidrogeradores

A vida moderna exige das famílias um uso crescente e perene da energia elétrica. Esta exigência também está presente nas indústrias e serviços devido à automatização e informatização de processos. A manutenção do fornecimento de energia é fundamental, com foco na continuidade e conformidade. Portanto, as centrais hidroelétricas devem ser capazes de fornecer energia com maior qualidade e menor custo, uma vez que o aumento da procura exige produção e transmissão de energia elétrica livre de perturbações [36].

As centrais hidroelétricas são a base da matriz energética brasileira. Os geradores hidroelétricos são os principais ativos industriais num sistema de geração de energia hidroelétrica. A ocorrência de falhas nesses hidrogeradores reduz a eficiência e pode interromper a geração de energia. A indisponibilidade do sistema energético exige altos custos de manutenção para o restabelecimento de ativos e multas impostas por órgãos reguladores, como a Agência Nacional de Energia Elétrica (ANEEL) no Brasil e a Entidade Reguladora dos Serviços Energéticos (ERSE) em Portugal. A seguir serão detalhados os três tipos de hidrogeradores mais aplicados em centrais hidroelétricas.

2.5.2 Turbinas Pelton

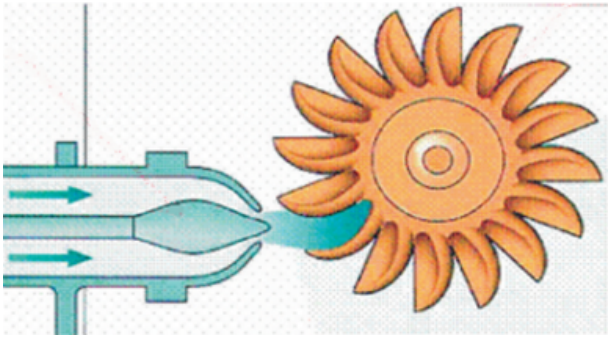
As turbinas Pelton são utilizadas em grandes quedas, assim, a energia é produzida pela alta velocidade dos jatos de água que incidem tangencialmente sobre suas pás [12, 59].

São normalmente preferidas para quedas maiores que 450 metros, porém podem ser usadas para quedas baixas como 200 metros. Na Figura 2.5 duas ilustrações desse modelo [12, 59].

2.5.3 Turbinas Kaplan

As unidades hidrogeradoras Kaplan são projetadas para operar onde uma pequena nascente de água está envolvida e suas turbinas podem ser usadas em locais com uma faixa de queda típica de 2 a 40 metros [60]. A Figura 2.6a ilustra uma UG do tipo Kaplan.

O ângulo (ou passo) das turbinas das lâminas pode ser alterado para se adequar ao

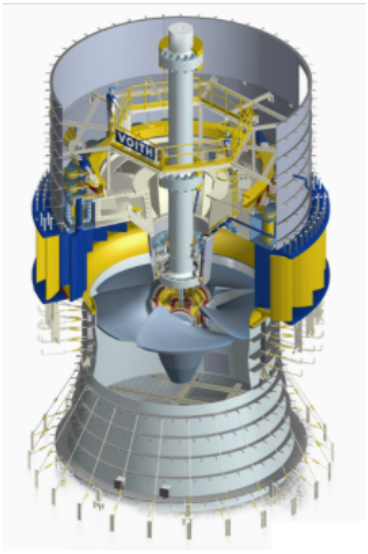


(a) Fluxo de uma turbina Pelton. [12].

(b) Turbina Pelton. [37]

Figura 2.5: Representações de uma turbina tipo Pelton.

fluxo de água. O recurso de passo ajustável das turbinas Kaplan permite que esses tipos de turbinas operem de forma eficiente em uma faixa mais ampla de queda d'água, permitindo variações no nível de água da barragem [80]. O hidrogerador Kaplan pode ser dividida em três sistemas principais: regulador de velocidade, sistema de turbina e eixo. A Figura 2.6b ilustra essas partes principais.



(a) Turbina Kaplan. [75].

(b) Componentes principais (hidrogerador Kaplan). [80]

Figura 2.6: Representações de uma turbina tipo Kaplan

O gerador é um componente da UG que é responsável por transformar a energia

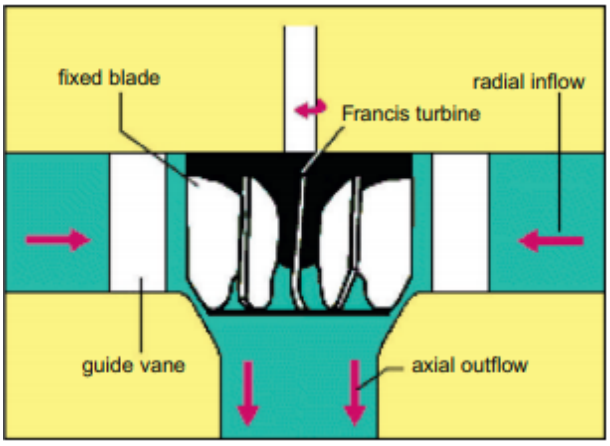
mecânica em elétrica. Em geral, é do tipo síncrono, isto é, a velocidade de rotação é constante e sincronizada com a frequência e a fase da tensão elétrica alternada. É composto pela parte fixa (estator) e pela parte rotativa (rotor), este, compõem-se do cubo com o eixo, que é acoplado diretamente ao eixo da turbina [32, 38].

O sistema de eixo é o elemento mecânico que interliga a turbina e o gerador. É responsável por proporcionar a rotação a ser transmitida ao eixo do gerador que é obtida pela conversão de energia hidráulica em mecânica pela turbina [32].

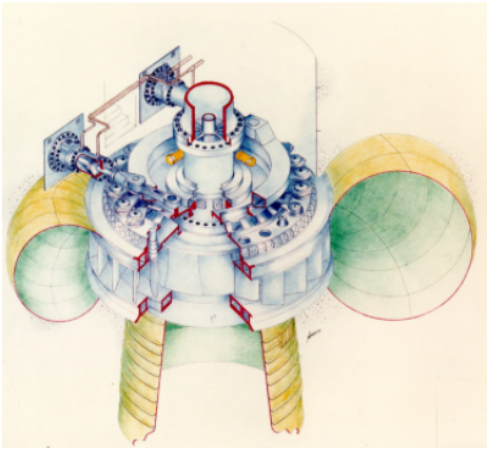
Já o regulador de velocidade tem como principal função manter a rotação da turbina constante para que o gerador forneça energia ao sistema elétrico, isso através da ação de controlar a abertura ou fechamento do distribuidor da turbina [45].

2.5.4 Turbina Francis

A sua principal característica é que a direção da água muda conforme passa pela turbina, ou seja, o fluxo inicialmente é radial e, após a passagem pela turbina torna-se axial [12]. A Figura 2.7a ilustra esse processo e a Figura 2.7b é uma representação de uma turbina Francis.



(a) Fluxo de uma turbina Francis. [12].



(b) Turbina Francis. [30]

Figura 2.7: Representações de uma turbina tipo Francis

Podem ser usadas para quedas de desde 3 metros até 600 metros, porém, sua melhor performance se encontra na faixa de 100 a 300 metros. Uma curiosidade é que as maiores

centrais hidroelétricas do mundo usam esse tipo de turbina, como a Central Hidroelétrica do Itaipu, as Três Gargantas na China. Já em Portugal existe a hidroelétrica do Foz-Tua instalada no Rio Tua que possui capacidade de 2×126 MW com duas turbinas do tipo Francis reversíveis [12, 10].

Capítulo 3

Métodos e técnicas

Neste capítulo é feita uma abordagem referente a linguagem de programação abordada, suas principais bibliotecas, os problemas de otimização baseados e os pacotes empregados na sua resolução.

3.1 Linguagem de programação

O uso de linguagens de programação tornou-se parte do quotidiano de ambientes industriais, sendo indispensáveis para a resolução de problemas desafiadores através da aplicação de algoritmos de *machine learning, big data* e IA, por exemplo [48].

Dentre elas, o *Python* tem-se destacado, com uma popularidade crescente, em especial, na ciência dos dados, um dos ramos da ciência da computação. É uma linguagem de programação de alto nível, isto é, dinâmica, interpretada, modular, multiplataforma e orientada a objetos, sendo amplamente aceita e aplicada no mercado por grandes empresas [57, 68].

[65] averiguou quais são as linguagens de programação mais usadas por investigadores na área de análise de dados, com grande destaque ao *Python*, como ilustrado na Figura 3.1.

Possui uma aprendizagem relativamente fácil, quando comparado as demais linguagens de programação, o que abrange o uso e aplicação para qualquer área de trabalho, não

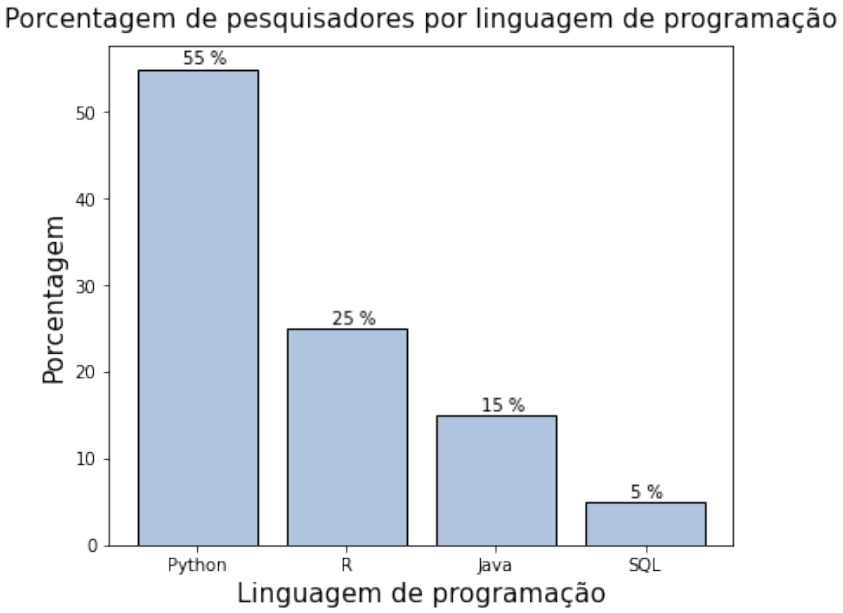


Figura 3.1: Percentagem de investigadores por linguagem de programação. [65].

apenas limitado a ciência da computação. Além disso, envolve menos codificação, isto é, linhas de código para desenvolver aplicações. Estima-se que o *Python* usa cerca de 1/5 de código em relação ao C, C++, *Java, Ruby* e *Javascript* para implementar a mesma lógica. O utilizador, consequentemente, ganha uma flexibilidade e eficiência maior na hora de programar, também facilitando a leitura e interpretação do código [48, 68].

Por fim, uma das maiores vantagens dessa linguagem é o fato de possuir um enorme ecossistema que consiste em bibliotecas que engloba todos os aspectos da ciência da computação, tornando-a muito poderosa e versátil e, também, por possuir comunidades de utilizadores espalhados pelo mundo dispostos a divulga-lá e auxiliar na resolução de problemas, sendo assim, a linguagem de programação escolhida para o desenvolvimento desse trabalho [48, 68].

3.2 Bibliotecas e problemas aplicados

As bibliotecas são um conjunto de módulos e funções cujo objetivo é simplificar diversos processos importantes como analisar e visualizar dados, criar modelos de *machine learning*, processar imagens entre outros. Dessa forma, permite que tarefas sejam automatizadas, reduz as linhas de códigos necessárias para implementar programas, aumentando a performance e flexibilidade do código [22, 57].

Segundo o *Python Package Index (PyPi)*, um repositório do *Python* em que é possível encontrar e instalar pacotes, existem mais de 200 mil bibliotecas que podem ser utilizadas. A seguir é explicado algumas das principais bibliotecas [21].

3.2.1 *NumPy*

O *NumPy*, é um pacote focado em computação científica, que tem como principal objetivo simplificar a implementação de cálculos numéricos complexos. Consiste em objetos, chamados de *arrays* (matrizes) multidimensionais, com uma variedade de funções e operações que podem ser aplicadas para álgebra linear, manipulação polinomial, operações estatísticas entre outras, algumas delas estão descritas na Tabela 3.1 [27, 46].

Tabela 3.1: Amostra de funções do *NumPy*. Adaptado de [27, 46].

Módulos	**Descrição**
np.array	Converter dados de entrada (lista, tuple, matriz, ou outro tipo de sequência) em uma matriz *NumPy*
np.empty	Cria uma matriz *NumPy* vazia
np.sqrt	Calcula a raíz quadrada
np.cos / np.sin / np.tan	Algumas das funções trigonométricas dessa biblioteca
np.log	Cálculos logaritmos
np.dot	Essa função realiza a multiplicação entre matrizes
np.inv	Calcula a matriz quadrada inversa
np.permutation	Retorna a permutação de uma sequência

É uma biblioteca amplamente utilizada por ser rápida, eficiente e prover alta performance na manipulação de matrizes multi-dimensionais. É empregada em outros pacotes como o *Pandas* e *SciPy*, além de ser frequentemente implementada armazenar dados de treinamentos de modelos de *machine learning* e no processamento de imagens [27, 46].

3.2.2 *SciPy*

O *SciPy* tem como objetivo problemas comuns da computação científica, fornecendo algoritmos para otimização, integração, interpolação, álgebra linear, equações diferencias e outros [74].

A Tabela 3.2 apresenta algumas funções e suas respectivas tarefas dessa biblioteca [46, 74].

Tabela 3.2: Amostra de módulos do *SciPy*. Adaptado de [46, 74].

Módulos	**Descrição**
scipy.integrate	Rotinas de integração numérica e equações diferenciais
scipy.linalg	Rotinas de álgebra linear
scipy.optimize	Otimização de funções
scipy.signal	Processamento de sinal
scipy.sparse	Matrizes esparsas e sistemas lineares esparsos
scipy.special	Implementação de funções matemáticas
scipy.stats	Contém funções estatísticas
scipy.weave	Ferramenta para usar o código C++ em linha para acelerar os cálculos da matriz

O *SciPy* também opera de forma eficiente em matrizes *NumPy*, de forma a trabalhar em conjunto com essa biblioteca, oferecendo rotinas amigáveis e eficientes [74].

3.2.3 *Pandas*

O nome *Pandas* vem de *panel data*, um termo da econometria para conjunto de dados estruturados. Apesar de ser inicialmente projetada para aplicação no setor financeiro, se tornou amplamente usada na análise de dados [46].

É um dos pacotes mais populares do *Python* por possuir uso e aprendizagem fácil, sendo utilizado para manipulação, preparação, limpeza e visualização de dados. Tem como grande vantagem poder trabalhar com diferentes tipos de dados, como Excel, CSV, HTML e SQL. Na Tabela 3.3 contém uma amostra de algumas de suas funções [48, 54].

Uma aplicação implementada por grandes empresas é para sistemas de recomendação. Um exemplo é a Netflix que utiliza os dados de seus clientes para sugerir sugestões de filmes, séries e documentários para seus clientes com base em suas preferências pessoais [22].

Tabela 3.3: Amostra de funções do *Pandas*. Adaptado de [46, 54].

Módulos	**Descrição**
pandas.read_csv / pandas.read_excel	Importam arquivos CSV e Excel, é possível importar outros tipos de arquivos além dos citados
DataFrame.head / DataFrame.tail	Usadas para visualizar as primeiras e últimas linhas de um dataset, respectivamente
DataFrame.describe	Apresenta um resumo estatístico do seus dados
DataFrame.sort_values	Altera a ordem dos valores em uma coluna classificando-a
DataFrame.fillna	Preenche os valores ausentes no dataset
DataFrame.groupby	Operação de agrupamento do *Pandas*. É possível aplicar funções a esses grupos, como por exemplo, agrupar o ID de sensores calculando a média das medições.
DataFrame.dropna	Remove linhas que contenham valores nulos
DataFrame.concat / DataFrame.merge	Funções usadas para combinar ou unir dados

3.2.4 *Matplotlib*

O *Matplotlib* é um pacote focado em gráficos, permitindo visualizações estáticas, animadas e interativas em *Python*. Com poucas linhas de código é possível gerar uma ampla variedade de gráficos como o de barras, pizza, histogramas, boxplots entre outros, proporcionando a identificação de padrões, correlações e tendências, favorecendo a tomada de decisões. A Tabela 3.4 apresenta algumas de suas funções [16, 22].

Tabela 3.4: Amostra de funções do *Matplotlib*. Adaptado de [16].

Módulos	**Descrição**
plt.figure	Cria uma figura
plt.subplots	Função usada para criar os eixos do gráfico
plt.set_title	Determina o título do gráfico
plt.set_xlabel / plt.set_ylabel	Nomeia os eixos x e y, respectivamente
plt.legend	Adiciona uma legenda ao gráfico
plt.grid	Configura as linhas de grade
plt.plot	Apresenta o gráfico
plt.savefig	Permite salvar os gráficos em um arquivo escolhido

Além disso, permite diversas possibilidades de customização gráfica, produz imagens com qualidade para publicações, exportação para diferentes formatos de arquivos e é uma extensão numérica para o *NumPy*, sendo a biblioteca mais popular na visualização de dados [16, 46].

3.2.5 *Scikit Learn*

É uma biblioteca voltada para construção de modelos de *machine learning*, com ferramentas simples e efetivas para efetuar análises preditivas. Conta com inúmeros algoritmos supervisionados e não supervisionados, como de regressão, classificação, cluterização, *random forest* e *k-means*, além de subpacotes que são utilizados para pré-processamento, treino e avaliação dos modelos criados, como apresentado na Tabela 3.5 [22, 40].

Tabela 3.5: Amostra de subpacotes do *Scikit Learn*. Adaptado de [40].

Nome	**Módulo**	**Tipo**	**Descrição**
Regressão linear	*sklearn.linear_model.LinearRegressio*	Algoritmo supervisionado	Modelo linear que procura prever o resultado de uma variável contínua baseado em outros dados históricos
Nearest Neighbor	*sklearn.neighbors.NearestNeighbors*	Algoritmo supervisionado	Procura classificar cada amostra de um conjunto de dados através da distância do seu vizinho mais próximo e, prever a variável alvo a partir disso
Clusterização	*sklearn.cluster*	Algoritmo não supervisionado	Agrupamento de dados similares em conjuntos distintos entre si
Restricted Boltzmann machines (RBM)	*neural_network.BernoulliRBM*	Algoritmo não supervisionado	É uma rede neural artificial baseada em um modelo probabilístico
Cross Validation	*sklearn.model_selection*	Seleção do modelo	É uma técnica de *machine learning* usada para avaliação de desempenho de modelos de aprendizado de máquina
Metrics and score	*sklearn.metrics*	Avaliação do modelo	Técnicas usadas para quantificar a qualidade do modelo criado
Pré processamento e transformadores	textitsklearn.preprocessing	Transformação dos dados	Conjunto de atividades que envolvem preparação, organização e estruturação dos dados, existindo diversos transformadores nesse subpacote

Foi iniciada em um projeto do *Google Summer of Code* em 2007, por ser projetada para interagir com outros pacotes como *NumPy*, *SciPy* e *Matplotlib* proporciona simplicidade e eficiência em suas diferentes aplicações sendo muito utilizada por profissionais da área

[40, 48].

3.3 Modelação matemática

O desenvolvimento do código deste trabalho foi inspirado em alguns problemas clássicos de otimização aliados a bibliotecas específicas do *Python*.

3.3.1 O Problema da Mochila

O Problema da Mochila é um problema clássico de otimização. Ele envolve um reservatório (mochila) com uma capacidade fixa W_i e um número de itens n como conteúdo. Cada item j com um respetivo peso $w_{i,j}$ e um valor v_j associado. O objetivo é maximizar o valor dos itens selecionados sem exceder a capacidade da mochila, como exemplificado a seguir através do problema de otimização (3.1) [62].

$$\begin{aligned} &\max \sum_{j=1}^{n} v_j x_j \\ &s.a. \sum_{j=1}^{n} w_{i,j} x_j \leq W_i \end{aligned} \tag{3.1}$$

Sendo x_j a variável binária que representa o item j, com $x_j \in \{0; 1\}$, $j \in \{1; \dots\ n\}$ e $i \in \{1; \dots\ m\}$. Assim, quando $x_j = 1$, significa que o item em questão foi selecionado e, consequentemente, $x_j = 0$, o item não foi considerado.

3.3.2 *Pyeasyga*

O *pyeasyga* é uma biblioteca direcionada para AG's. É caracterizada por possuir uma implementação simples e fácil, sem necessitar de conhecimentos especializados em ciências da computação. Para a aplicação dele, os seguintes passos devem ser executados [61].

1. Importar o módulo

```
from pyeasyga import pyeasyga
```

2. Carregar e configurar o *dataset*

3. Executa o AG do pacote, chamado *GeneticAlgorithm*, este sendo armazenado em uma variável, neste caso chamada ga. É obrigatório informar o *dataset*

```
ga = pyeasyga.GeneticAlgorithm(dataset)
```

   É possível utilizar alguns parâmetros opcionais que podem resultar em um algoritmo melhor.

```
ga = pyeasyga.GeneticAlgorithm(dataset,
                               population_size,
                               generations,
                               crossover_probability,
                               mutation_probability,
                               elistism,
                               maximise_function)
```

4. Definir a função desempenho para o AG, com ela possuindo dois parâmetros, o *indivíduo*, que representa um possível candidato a solução e o próprio *dataset*

5. Atribuir a função desempenho a *function_fitness*

```
ga.fitness_function = funcao_desempenho_definida
```

6. Executar o algoritmo

```
ga.run()
```

7. Apresentar o resultado, isto é, a melhor solução

```
print(ga.best_individual())
```

Opcionalmente é possível criar os indíviduos, definindo uma função que se adeque ao problema em questão, assim como, definir os operadores genéticos do AG (cruzamento, mutação e seleção) [61].

3.3.3 O Problema do Caixeiro Viajante

O dilema do Caixeiro Viajante é um problema clássico no âmbito de otimização combinatória. Ele consiste em um conjunto de cidades com suas respetivas distâncias e o principal objetivo deste problema é determinar a ordem pela qual o vendedor deve percorrer as cidades afim de reduzir a distância visitada. É um problema que pode ser adaptado para diversas áreas de estudo como informática, programação logística e engenharia industrial [20, 26].

A formulação do problema é dada um número de cidades n, com $n \in Z = \{1, 2, \ldots\}$, cada uma com uma distância associada, $c_{i,j}$, entre duas cidades i e j, com $i \in \{1; \ldots\ n\}$ e $j \in \{1; \ldots\ n\}$, a variável binária $x_{i,j}$ representa a rota que o vendedor está seguindo da cidade i para a cidade j, assim temos a seguinte equação (3.2):

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} x_{i,j} \tag{3.2}$$

Sujeito as seguintes restrições:

$$\sum_{j=1, i \neq j}^{n} x_{i,j} = \sum_{i=1, i \neq j}^{n} x_{i,j} = 1 \tag{3.3}$$

A restrição (3.3) garante que cada cidade é visitada apenas uma vez.

$$x_j \in \{0; 1\} \tag{3.4}$$

A equação (3.4) por sua vez atribui valor 0 para rotas não selecionadas e 1 caso contrário

$$\sum_{i; j \in S} x_{i,j} \leq |S| - 1 \tag{3.5}$$

Por fim, a fórmula (3.5) representa que se o vendedor deixa uma cidade, ele não pode retornar a ela mais tarde. Esta restrição assegura que nenhum ciclo desconectado seja formado como uma solução, sendo S o número de nós, com $S \in \{2, \ldots, n-2\}$.

Para esse problema foram utilizadas diferentes bibliotecas do *Python* afim de comparação, são eles: pymoo, py2opt e tsp_solver.

3.3.4 *Pymoo*

O *pymoo* é é um *framework* de otimização de mono ou multi-objetivo com diversos algoritmos implementados. Para este trabalho foi implementado um AG próprio do pacote, que como explicado anteriormente é um eficiente algoritmo para resolução de problemas de otimização, que também é o caso do caixeiro viajante. Sua implementação se dá através dos seguintes passos: [8, 9].

1. Importar os módulos

```
from pymoo.algorithms.soo.nonconvex.ga import GA
from pymoo.optimize import minimize
from pymoo.problems.single.traveling_salesman
import create_random_tsp_problem
from pymoo.operators.sampling.rnd import PermutationRandomSampling
from pymoo.operators.crossover.ox import OrderCrossover
from pymoo.operators.mutation.inversion import InversionMutation
from pymoo.termination.default
import DefaultSingleObjectiveTermination
from pymoo.core.repair import Repair
import numpy as np
```

2. Carregar e configurar o *dataset*

3. Definir o problema do Caixeiro Viajante

4. Para evitar que a permutação dos dados inicie com um número arbitrário é recomendado definir a classe *Repair* de forma a garantir que o algoritmo inicie a partir do primeiro elemento com índice 0, a própria biblioteca demonstra como cria-lá.

```
class StartFromZeroRepair(Repair):
    def _do(self, problem, X, **kwargs):
```

```
I = np.where(X == 0)[1]
for k in range(len(X)):
    i = I[k]
    X[k] = np.concatenate([X[k, i:], X[k, :i]])
return X
```

5. Executa o AG da biblioteca, este sendo armazenado em uma variável, neste caso chamada algoritmo. Possui os seguintes parâmetros:

```
algoritmo = GA(pop\_size=20,
               sampling=PermutationRandomSampling(),
               mutation=InversionMutation(),
               crossover=OrderCrossover(),
               repair=StartFromZeroRepair(),
               eliminate_duplicates=True
```

6. Atribuir um critério de paragem do algoritmo, por exemplo, 200 gerações. A condição de terminação define, quando o loop do algoritmo termina, assim, caso não exista melhoras nas últimas 200 gerações, o algoritmo irá finalizar sua execução.

```
terminacao = DefaultSingleObjectiveTermination(period=200,
                                               n_max_gen=np.inf)
```

7. Executar o algoritmo

```
resultados = minimize()
```

8. Verificar o resultado, isto é, a melhor solução

```
print(resultado.X)
```

3.3.5 *py2opt*

O *py2opt*, por sua vez, é um pacote focado apenas na resolução do problema do caixeiro viajante. Ele utiliza o 2-opt heurístico, que é um algoritmo relativamente simples, mas que retorna boas soluções para o problema em questão. Entretanto, como ressalta a própria biblioteca, não há uma garantia de encontrar o ótimo global, portanto os resultados podem variar a cada iteração. As etapas para executá-lo estão a seguir: [3, 28].

1. Importar o módulo

```
from py2opt.routefinder import RouteFinder
```

2. Carregar e configurar o *dataset*, sendo necessário criar duas listas, uma com os índices das máquinas, *ID* e outra com os dados da matriz de distâncias, *dist_mat* com as respetivas distâncias, como exemplificado a seguir:

```
ID = ['A', 'B', 'C', 'D']
dist_mat = [[0, 29, 15, 35],
            [29, 0, 57, 42],
            [15, 57, 0, 61],
            [35, 42, 61, 0]]
```

3. Criar uma instância da classe RouterFinder, informando as listas definidas acima e o número de interações desejadas.

```
route_finder = RouteFinder(dist_mat,
                           ID,
                           iterations=5)
```

4. Executa o método de resolução do algoritmo

```
melhor_distancia, melhor_rota = route_finder.solve()
```

5. Apresenta o resultado, isto é, a melhor solução

```
print(melhor_distancia, melhor_rota)
```

3.3.6 *tsp_solver*

O *tsp_solver* também foi desenvolvido especificamente para o do problema do caixeiro viajante. Ele utiliza um algoritmo ganancioso, que também apresenta uma solução local ótima, e mesmo, essa biblioteca possuindo uma otimização, ainda sim não há garantia de uma solução global ótima, sua implementação está exemplificada a seguir: [67].

1. Importar os módulos

```
from tsp_solver.greedy import solve_tsp
from tsp_solver.util import path_cost
```

2. Carregar e configurar o *dataset*, sendo necessário criar uma lista com os dados da matriz de distâncias, *dist_mat* com as respetivas distâncias, como exemplificado a seguir:

```
dist_mat = [[0, 29, 15, 35],
            [29, 0, 57, 42],
            [15, 57, 0, 61],
            [35, 42, 61, 0]]
```

3. Executa o método de resolução do algoritmo

```
melhor_rota = solve_tsp(dist_mat)
melhor_distancia = path_cost(dist_mat, melhor_rota)
```

4. Apresentar o resultado, isto é, a melhor solução

```
print(melhor_distancia, melhor_rota)
```

Capítulo 4

Caso de estudo

Neste capítulo é apresentada com algum detalhe a central hidroelétrica que inspirou o estudo, assim como os dados utilizados para a resolução do projeto. Na abordagem dos problemas são demonstrados as formulações matemáticas e suas respectivas restrições, visando tornar a aplicação dos algoritmos adaptados a uma realidade industrial.

4.1 Caracterização da Central Hidroelétrica

Este estudo foi inspirado em uma central hidroelétrica localizada na região norte do Brasil que possui três hidrogeradores do tipo Kaplan, cada um operando 166.25 Megawatt (MW), com uma capacidade instalada de aproximadamente 500 MW. Uma árvore funcional simplificada, focada nos três principais sistemas mecânicos do hidrogerador, o eixo, o regulador de velocidade e a turbina, com os seus principais respectivos subsistemas, é mostrada na Figura 4.1 [2, 6, 7].

A árvore funcional apresenta, em uma abordagem sistemática, as inter-relações entre os componentes do sistema em questão de forma lógica e hierárquica, sendo uma forma visual para entender a organização dos equipamentos e componentes de cada subsistema presente na indústria. É uma ferramenta poderosa para o setor da manutenção, já que esse diagrama pode ser usado para estudar a confiabilidade de sistemas complexos e, posteriormente, entender os modos de falhas para então estruturar um plano de manutenção

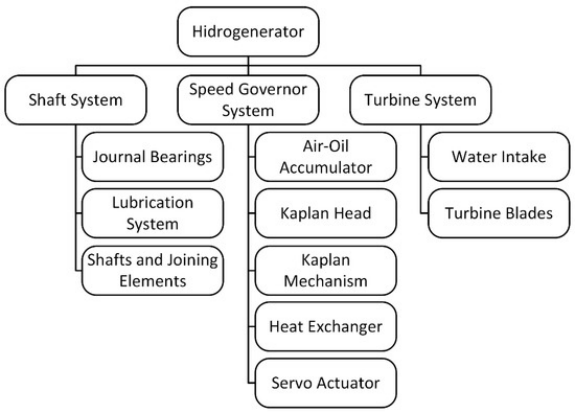


Figura 4.1: Árvore funcional simplificada. [2].

a partir dos componentes críticos. Os dados recolhidos são provenientes de projetos de investigação desenvolvido na Universidade de São Paulo (USP) (Brasil) em cooperação técnica com a *Universitá Politecnica Delle Marche (UNVPM)* (Itália) [2, 6, 7].

4.2 Seleção das atividades de manutenção

Um dos objetivos do algoritmo é otimizar a seleção das atividades a serem realizadas pelos técnicos do setor da manutenção, proporcionando uma agilidade maior no planeamento da manutenção.

4.2.1 Dados

Os dados utilizados no estudo, referidos na Subsecção 4.1 são referentes às atividades de uma central hidroelétrica com três hidrogeradores. No conjunto de dados utilizados, representados pela Tabela 4.1, verifica-se que existem quatro grandes subsistemas, sendo eles o gerador, com três componentes (painel, sistema de monitorização e mancal escora) e quatro atividades, em seguida a excitatriz, turbina e regulador de velocidade, todos com uma atividade. Em geral, essas tarefas são de limpeza, inspeção ou lubrificação.

O *dataset* contém sete linhas e dez colunas e estão descritos na Tabela 4.1, em que é apresentado a identificação da atividade (ID), representado por um número inteiro e

único, o componente da central em que a atividade será realizada (Comp.), o número de dias em que não é realizado essa atividade de manutenção no respectivo componente (Data exec.), com uma média de 42731 dias, a descrição da atividade a ser realizada pelo técnico (Descrição da atividade).

A classificação da máquina (Class.) aponta a criticidade do elemento em questão, separando os componentes com uma importância maior sobre a Central Hidroelétrica, requisitando um maior cuidado no planeamento das atividades de manutenção, cuja moda é de 20. Os parâmetros de classificação adotados pela central em estudo são divididos em A, B, C e D, com os respectivos valores númericos 30, 20, 10 e 0.

Tabela 4.1: Amostra dos dados utilizados.

ID	**Comp.**	**Data exec.**	**Descrição da atividade**	**Class.**	**D.E**	**C.O.**	**Freq.**	**Clus.**	**N. Tec.**
1	Painel do gerador	42455	Limpeza do painel do sistema de monitorização dos geradores	20	2	0	35040	3	2
2	Painel do gerador	42483	Inspeção no painel	20	4	1	26280	5	2
3	Excitatriz	43545	Inspeção e manutenção nas escovas	30	6	0	4320	2	4
4	Painel da turbina	42823	Inspeção no painel de comando para manutenção do exaustor do poço da turbina	20	2	1	5000	8	2
5	Motor da bomba do regulador de velocidade	42117	Lubrificação no motor da bomba	20	2	1	5000	8	2
6	Instrumento de Ensaio Eletroeletrônico	42872	Inspeção operacional do sistema de monitorização dos geradores	0	2	1	17520	11	2
7	Mancal escora do gerador	42823	Inspeção na caixa de instrumentos do reservatório do mancal escora	30	2	0	17520	2	2

Já a duração estimada que atividade leva para ser concluída em horas (D.E.), neste *dataset* os valores variam entre 2 e 6, com uma mediana e moda de 2 horas, a condição

de operação (C.O.) é uma coluna de carater binário, sendo 1 representando que a atividade em questão pode ser realizada com a central hidroelétrica em operação e 0 indica que a unidade geradora de energia precisa estar em paragem para realizar a tarefa de manutenção, com quatro atividades classificadas como 1 e três como 0.

A coluna "Freq."representa a periodicidade necessária para a manutenção desta máquina em dias, com uma média de 733 dias, a prioridade de execução do serviço (Clus.), foi determinada pela empresa representando uma categorização das atividades em um intervalo de 1 a 11, com o primeiro grupo contendo tarefas mais críticas e assim por diante, e, finalmente, o número de técnicos necessários para realizar as tarefas (N. Tec.) e apresenta uma moda e mediana de 2 técnicos por atividade.

4.2.2 Parâmetros de entrada do problema

Para que o problema se adapte a uma realidade industrial é necessário informar ao algoritmo alguns parâmetros iniciais para que a função objetivo realize os cálculos necessários. A Figura 4.2 apresenta os parâmetros que são necessários ser informados pelo utilizados do programa.

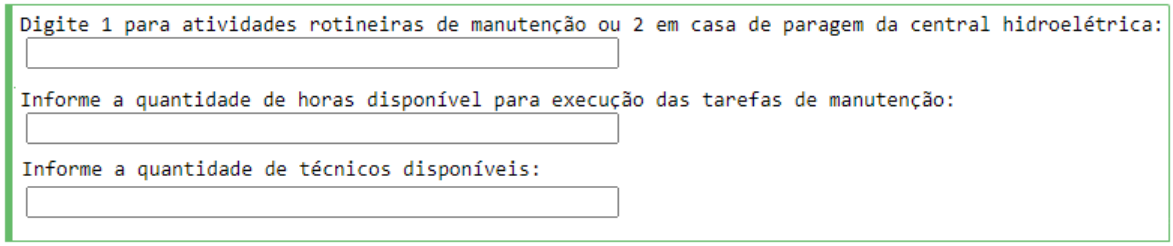


Figura 4.2: Visualização dos parâmetros de entrada.

- *Tipo de rota*: tem por objetivo explicitar se são atividades rotineiras de manutenção ou se é uma paragem total da central hidroelétrica, dessa forma a prioridade das atividades é ajustada.

- *Tempo disponível para execução das atividades*: é a carga horária total de trabalho dos técnicos para executar as tarefas selecionadas.

- *Número de técnicos disponível*: é a carga de trabalho que a central possui no momento.

4.2.3 Modelo matemático

A função objetivo para seleção das atividades de manutenção pretende maximizar a quantidade de tarefas a serem desempenhadas no tempo estimado, levando em consideração parâmetros como a criticidade da máquina. Esta é responsável por fazer a corresponder a cada solução um número real, o que permite comparar e avaliar as diferentes soluções do problema.

O valor ótimo para a função objetivo deste problema é obtido pela maximização do problema envolvendo a quantidade de atividades selecionadas, n, sem exceder o tempo disponível de trabalho levando em consideração a condição operativa da central hidroelétrica, isto é, se a mesma está em paragem ou não. Em caso de paragem, o cálculo é de acordo com a equação (4.1)

$$\max \sum_{i=1}^{n} (\frac{(\frac{DE_i}{F_i}) + 3C_i + 5K_i}{100}) + 1 \tag{4.1}$$

Em que: DE_i é a quantidade de dias que a atividade não é executada, F_i a periodicidade indicada em que a tarefa deve ser realizada em dias, C_i a criticidade do componente, K_i é o respectivo cluster, i é o índice da atividade selecionada. É atribuído um peso para cada variável determinados pela própria central e, uma vez que as paragens das centrais ocorrem em períodos pré determinados e as atividades prioritárias devem ser aquelas que exigem que que a unidade geradora esteja em paragem, por esse motivo é somado o valor 1 ao final da equação (4.1) para atribuir um peso maior a mesma.

O cálculo $\frac{DE_i}{F_i}$ tem por objetivo verificar quais atividades estão a mais tempo sem serem executadas, possuindo uma prioridade maior na função objetivo.

No caso do tipo de rotar selecionado for de atividades rotineiras a diferença entre as equações (4.1) e (4.2) se dá no acréscimo de uma unidade, além do fato de que tarefas que exigem paragem da unidade geradora não podem ser escolhidas.

$$\max \sum_{i=1}^{n} \left(\frac{\left(\frac{DE_i}{F_i}\right) + 3C_i + 5K_i}{100}\right) \tag{4.2}$$

O grande objetivo das funções desse tópico é obter um vetor de caráter binário, em que 1 representa que a atividade foi selecionada e 0 não, a solução ótima é tida como a maximização da quantidade de atividades presentes nesse vetor, respeitando as restrições impostas pela central. Posteriormente as atividades selecionadas nesse vetor serão utilizadas para definir a melhor rota de execução das tarefas.

Este estudo levou em consideração algumas restrições afim de tornar a aplicação do algoritmo em condições industriais reais. A primeira delas envolve a coluna C.O. da Tabela 4.1 em que representa se a tarefa é uma atividade rotineira de manutenção, isto é, não necessita parar a máquina, podendo interromper o processo de geração de energia. São tarefas que se enquadram como manutenção preventiva, como inspeções instrumentais, lubrificações e limpezas, ou manutenção preditiva no caso de medições de vibração de motores, por exemplo.

Caso seja uma paragem programada de manutenção da central hidroelétrica, o algoritmo priorizará as atividades que necessitam que a máquina esteja em paragem para que os técnicos realizarem as tarefas adequadas que, normalmente, são mais desafiadoras, exigem mais tempo, trocas de peças e um cuidado maior.

Uma outra restrição é que o tempo das tarefas selecionadas não ultrapassem o tempo total de atividade ou paragem, de forma a tornar o planeamento da manutenção mais assertivo. Por fim, o algoritmo leva em consideração o número de técnicos recomendados para executar a atividade, para que não seja atribuído um número inferior de técnicos, o que resultaria em retrabalho ou mais tempo que o de fato necessário para desempenhar aquela função, ou ociosidade no caso de enviar um grupo de trabalho maior que o exigido.

4.3 Otimização das rotas de manutenção

Uma vez que as tarefas foram selecionadas e atribuídas a cada técnico, o próximo passo é determinar a melhor rota que cada técnico deve seguir, de forma a aumentar o desempenho da equipa, otimizando o tempo de trabalho, evitando deslocamentos desnecessários, elevando o rendimento não apenas dos trabalhadores, mas da central como um todo.

4.3.1 Dados

Os dados utilizados para otimização da rota estão descritos na Tabela 4.2, os valores são inspirados em dados reais, porém não são totalmente validados.

Tabela 4.2: Matriz da estimativa de deslocamentos entre as máquinas, em minutos.

ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	9	4	10	8	15	3	9
1	9	0	9	1	13	6	8	14
2	4	9	0	14	12	19	7	13
3	10	1	14	0	14	5	9	15
4	8	13	12	14	0	23	11	17
5	15	6	19	5	23	0	14	20
6	3	8	7	9	11	14	0	12
7	9	14	13	15	17	20	12	0

A Tabela 4.2 é uma matriz com o tempo de deslocamento, em minutos, em que cada coluna e linha representa um componente da central hidroelétrica, identificado por um número de identificação, sendo o *ID* 0 o ponto de partida dos trabalhadores.

4.3.2 Modelo matemático

A função objetivo para seleção das rotas de manutenção pretende minimizar o tempo gasto por cada técnico no caminho percorrido. Esta é responsável por atribuir um valor para cada rota a partir de todas as combinações possíveis de caminhos, o que permite comparar e avaliar as diferentes soluções do problema.

A minimização do tempo gasto em deslocamento é considerado o valor ótimo para esta função, sendo obtida pela soma do valor do tempo entre a primeira máquina presente na

rota e a seguinte máquina indicada na rota.

Dada uma matriz com os tempos de deslocamento D_{nxn}, em que n é o número de atividades presentes na rota, o cálculo é acordo com a equação (4.3)

$$\min \sum_{i;j \in A}^{n} d_{i,j} \tag{4.3}$$

Em que i e j indicam a máquina em questão, A representa o conjunto das atividades selecionadas anteriormente descrito na Subseção 4.2, $d_{i,j}$ é o item da matriz D_{nxn}, cujo valor representa o tempo de deslocamento entre uma máquina e a próxima da rota.

Para este problema existem algumas restrições de forma a tornar a aplicação do algoritmo coerente e assertiva. A primeira delas é que cada máquina é visitada uma única vez, visto que uma vez que o manutentor está trabalhando em uma determinada máquina deve realizar todos os procedimentos necessários nela para então prosseguir para o próximo equipamento.

Uma outra restrição é a consideração de um ponto de partida inicial dos técnicos, denotado pelo índice 0, pois usualmente o expediente de trabalho em uma indústria ou central inicia-se a partir da sala de permanências.

4.4 Pressupostos simplificativos

Para o desenvolvimento do problema foram consideradas algumas simplificações.

Considerou-se que não há diferenciação entre as especialidades dos técnicos de manutenção, pois não havia informações suficientes nos dados disponíveis que apontasse a quantidade de técnicos de cada departamento necessários para realizar as tarefas em questão.

Foram desprezados eventuais custos relacionados as atividades, assim como stock de materiais, por exemplo, caso haja a necessidade de trocar algum componente de um subsistema e o mesmo estar em falta, sendo necessário postergar a tarefa, ou uma determinada ação possuir um gasto elevado estando fora do orçamento disponível do setor.

Como mencionado anteriormente, é considerado um ponto de partida inicial das rotas de manutenção e os valores usados são inspirados em dados reais, uma vez que não haviam informações suficientes para estabelecer a localização precisa de cada maquinário na central.

Capítulo 5

Resultados e discussão

Este capítulo apresenta os testes realizados para verificar que o projeto desenvolvido cumpre os objetivos assumidos e resolve, de facto, o problema descrito no Capítulo 4. A acompanhar os resultados das diversas simulações encontram-se descritas as alterações efetuadas e as respetivas interpretações.

5.1 Descrição dos casos simulados

Este estudo procura otimizar duas áreas do planeamento do setor de manutenção de uma central hidroelétrica, a primeira através da aplicação de um algoritmo genético (AG), com o pacote *pyeasyga*, para identificar, selecionar e atribuir tarefas do setor aos técnicos de manutenção. A seguinte é referente a otimização do caminho percorrido por cada trabalhador na execução desse planeamento gerado, para tal, foram aplicados três bibliotecas do *Python* cada um com algoritmos diferentes, sendo eles: 2-opt heurístico (*py2opt*), um algoritmo genérico com uma abordagem gananciosa (*tsp_solver)* e AG (*pymoo*) com a finalidade de comparar os resultados obtidos e definir a melhor abordagem para a resolução do problema proposto.

5.2 Simulação dos parâmetros do AG

Um AG pode ser aplicado a uma ampla variedade de problemas de otimização, contudo os parâmetros do algoritmo devem ser ajustados para cada caso. Assim, deve-se especificar critérios como o número da população inicial, a quantidade de gerações que o algoritmo irá gerar, a taxa dos operadores genéticos (cruzamento e mutação) e se há elistismo na reprodução ou não.

A seleção adequada desses parâmetros pode melhorar o desempenho do problema, definindo uma combinação assertiva entre eles é possível obter uma melhoria em propriedades como o tempo de computação, a robustez e a qualidade da solução.

O objetivo principal da simulação a seguir é averiguar a viabilidade da aplicação de um AG para a otimização da seleção de atividades de manutenção de centrais hidroelétricas, determinando quais parâmetros do AG são mais adequados para o problema, avaliando a qualidade da solução e a viabilidade temporal do programa desenvolvido.

Para testar a abordagem foi considerado que a unidade geradora continua em operação, ou seja, as tarefas a serem escolhidas devem ser aquelas rotineiras em que as máquinas não precisam estar em paragem. Foi definido uma duração máxima de 10 horas de atividade e uma carga de trabalho de 3 técnicos disponíveis para realizar as tarefas de manutenção.

Tabela 5.1: Melhor solução esperada pelo AG

Pontuação	**Melhor solução**	**Atividades selecionadas**
6.733	[0, 1, 0, 1, 1, 1, 0]	[2,4,5,6]

A Tabela 5.1 cujas colunas representam o melhor valor obtido pela função de aptidão (Pontuação), a melhor solução apresentada pelo algoritmo genético, dentro do parênteses reto e com caráter binário, cujo valor "1"indica que a atividade foi selecionada e "0"não. (Melhor solução) e os respectivos ID's das atividades selecionadas (Atividades selecionadas). Tem por objetivo ilustrar os valores tomados como a melhor solução do algoritmo desenvolvido, utilizadas para averiguar a taxa de acerto das combinações de parâmetros do AG. Para cada um desses critérios foi definido um intervalo de valores para guiar os testes, são eles:

- *Tamanho da população* = 100 ou 200;
- *Número de gerações* = 30 ou 60;
- *Probabilidade de cruzamento* = 0.8 ou 0.9;
- *Probabilidade de mutação* = 0.05 ou 0.1;
- *Elitismo* = Falso ou Verdadeiro.

5.2.1 Simulação 1: sem elitismo

A reprodução elitista dentro de um AG procura garantir que que os melhores indivíduos dentro da população não desapareçam após a aplicação dos operadores genéticos, sendo garantidos na próxima geração. O resultado a seguir considera que não há elitismo afim de testar o comportamento do programa sem esse parâmetro na procura da melhor solução.

A biblioteca implementada foi o *pyeasyga* com os parâmetros do AG testados e comparados com os seguintes valores sumarizados na Tabela 5.1, em que é apresentado o tamanho da população de indivíduos (Tamanho Pop.), o número de gerações (Ger.), a probabilidade de cruzamento e mutação, respectivamente, (Cruzamento prob. e Mutação prob.), o tempo médio de execução do algoritmo (Tempo médio), o desvio padrão do tempo (Desvio) e a taxa de acerto do algoritmo (Taxa) em que compara o percentual de resultados de cada execução com a melhor solução esperada com os dados de entradas escolhidos.

Verifica-se que com esses parâmetros o algoritmo não transmite a confiabilidade que o planeamento da manutenção de uma central hidroelétrica exige, pois a taxa de acerto está baixa, não ultrapassando um percentual de 70%. Uma má execução desse setor pode causar a falha de equipamentos podendo causar interrupções da geração de energia do sistema hidroelétrica, gerando inúmeras consequências [13, 15].

Tabela 5.2: Comparação entre os parâmetros do AG sem elistismo

Tamanho Pop.	**Ger.**	**Cruzamento prob.**	**Mutação prob.**	**Tempo médio**	**Desvio**	**Taxa de acerto**
100	30	0.8	0.05	0.070	0.009	60%
100	30	0.8	0.1	0.074	0.016	10%
100	30	0.9	0.05	0.071	0.013	20%
100	30	0.9	0.1	0.065	0.006	40%
100	60	0.8	0.05	0.107	0.005	40%
100	60	0.8	0.1	0.128	0.017	30%
100	60	0.9	0.05	0.131	0.014	30%
100	60	0.9	0.1	0.131	0.015	30%
200	30	0.8	0.05	0.126	0.008	40%
200	30	0.8	0.1	0.112	0.007	70%
200	30	0.9	0.05	0.107	0.002	50%
200	30	0.9	0.1	0.109	0.006	60%
200	60	0.8	0.05	0.217	0.009	40%
200	60	0.8	0.1	0.207	0.003	40%
200	60	0.9	0.05	0.212	0.008	50%
200	60	0.9	0.1	0.214	0.004	50%

5.2.2 Simulação 2: com elistismo

Para essa simulação foram considerados os mesmos dados de entrada com a diferença de que o AG possui uma reprodução elitista.

Tabela 5.3: Comparação entre os parâmetros do AG com elistismo

Tamanho Pop.	**Ger.**	**Cruzamento prob.**	**Mutação prob.**	**Tempo médio**	**Desvio**	**Taxa de acerto**
100	30	0.8	0.05	0.058	0.007	100%
100	30	0.8	0.1	0.055	0.002	100%
100	30	0.9	0.05	0.057	0.005	100%
100	30	0.9	0.1	0.057	0.007	100%
100	60	0.8	0.05	0.116	0.013	100%
100	60	0.8	0.1	0.129	0.019	100%
100	60	0.9	0.05	0.119	0.021	100%
100	60	0.9	0.1	0.121	0.016	100%
200	30	0.8	0.05	0.114	0.015	100%
200	30	0.8	0.1	0.119	0.022	100%
200	30	0.9	0.05	0.128	0.026	100%
200	30	0.9	0.1	0.125	0.027	100%
200	60	0.8	0.05	0.239	0.019	100%
200	60	0.8	0.1	0.238	0.018	100%
200	60	0.9	0.05	0.241	0.029	100%
200	60	0.9	0.1	0.245	0.028	100%

A Simulação 2, contida na Tabela 5.3 evidencia a importância de testar a combinação

entre os parâmetros do programa para obter o melhor desempenho possível. É notório o impacto que a uma reprodução elitista tem em um AG, uma vez que existe uma garantia que os melhores indivíduos se mantenham nas gerações seguintes.

Uma vez que a taxa de acerto atende as expectativas, isto é, tem uma frequência de 100%, os critérios para analisar e definir a melhor combinação dos parâmetros restantes são o tempo médio de execução e o desvio padrão. Verifica-se que os parâmetros tamanho da população e número de gerações influenciam mais no tempo de execução que as demais, o que é esperado, já que esses parâmetros aumentam o número de indivíduos e gerações do AG, aumentando o período de execução do programa. Por outro lado, as probabilidades de cruzamento e mutação não apresentam um padrão no aumento ou diminuição do tempo. Por possuírem um tempo de execução menor, foram selecionados os quatros primeiros resultados, representados pela Tabela 5.4.

Tabela 5.4: Análise dos resultados da Simulação 2

ID	**Tamanho Pop.**	**Ger.**	**Cruzamento prob.**	**Mutação prob.**	**Tempo médio**	**Desvio**	**Intervalo**
1	100	30	0.8	0.05	0.058	0.007	[0.051, 0.065]
2	100	30	0.8	0.1	0.055	0.002	[0.053, 0.057]
3	100	30	0.9	0.05	0.057	0.005	[0.052, 0.062]
4	100	30	0.9	0.1	0.057	0.007	[0.051, 0.064]

Um dos objetivos de um programa é que sua execução seja a mais rápida possível, de forma a disponibilizar os resultados o quanto antes aos utilizadores. Assim, para selecionar a melhor combinação de parâmetros foi na Tabela 5.4 identificado o ID número 2 com o intervalo com menor tempo de execução necessário. Portanto, o primeiro algoritmo desenvolvido, cujo foco é a seleção de atividades, a melhor combinação é dada por:

- *Tamanho da população* = 100;
- *Número de gerações* = 30;
- *Probabilidade de cruzamento* = 0.8;
- *Probabilidade de mutação* = 0.1;
- *Elitismo* = *True*;

5.2.3 Simulação 3: paragem da central hidroelétrica

Um planeamento eficaz de manutenção deve conter paragens periódicas e programadas da planta industrial, pois existem diversas tarefas que apenas podem ser efetuadas com o equipamento sem estar operação. A simulação 3 procura verificar se o procedimento definido atribuirá as atividades corretamente utilizando os parâmetros do AG definidos anteriormente, maximizando a quantidade de serviço dentro da duração pré-estabelecida e o número de colaboradores. A Tabela 5.5 apresenta as tarefas que necessitam da paragem do sistema:

Tabela 5.5: Atividades que exigem paragem da planta hidroelétrica.

ID	**Componente**	**Descrição da atividade**	**Duração** estimada	**C.O.**	**N. Tec.**
1	Painel do gerador	Limpeza do painel do sistema de monitorização dos geradores	2	0	2
3	Excitatriz	Inspeção e manutenção nas escovas	6	0	4
7	Mancal escora do gerador	Mancal escora do gerador	2	0	2

O teste da abordagem foi similar ao anterior com a diferença de que foi considerado a interrupção da geração de energia, ou seja, o sistema hidrelétrico está em paragem. Da mesma forma, foi definido uma duração máxima de 10 horas de atividade e uma carga de trabalho de 3 técnicos disponíveis para realizar as tarefas de manutenção. Assim, é esperado que o algoritmo gere um planeamento contendo as atividades com ID 1 e 7, uma vez que a tarefa com ID 3 exige um número maior de técnicos do que de fato há disponível. A Tabela 5.6 apresenta o resultado do algoritmo com os parâmetros definidos de acordo com as conclusões obtidas da simulação anterior:

Tabela 5.6: Melhor solução esperada pelo AG

Pontuação	**Melhor solução**	**Atividades selecionadas**
10.122	[1, 0, 0, 1, 1, 1, 1]	[1,4,5,6,7]

Portanto, o programa cumpre com os objetivos estabelecidos, visto que atribuiu corretamente as tarefas sem ultrapassar o limite de colaboradores disponíveis e maximizando

a quantidade de atividades selecionadas de forma a preencher o total de horas determinadas. No Anexo A contém todos os testes realizados com as combinações de parâmetros do AG.

5.3 Simulação dos algoritmos de otimização de rota

A abordagem dessa simulação considera duas situações, sendo a primeira que a unidade geradora continua em operação, ou seja, as tarefas a serem escolhidas devem ser aquelas rotineiras em que as máquinas não precisam estar em paragem, a outra considera que o processo de geração de energia da central hidroelétrica foi interrompido, ou seja, houve uma paragem total da central. Para ambos foi definido uma duração máxima de 10 horas de atividade e uma carga de trabalho de 3 técnicos disponíveis para realizar as tarefas de manutenção.

5.3.1 Simulação 4: atividades rotineiras de manutenção

Neste estudo considera-se como uma atividade rotineira aquela em que não há necessidade de interromper o funcionamento da máquina, ou seja, a tarefa ocorre com o sistema hidroelétrica em operação. Alguns exemplos dessas atribuições são inspeções nos painéis de comando, afim de identificar alguma possível avaria e lubrificação de motores, reduzindo o atrito e desgastes entre os seus componentes, prologando a vida útil desse equipamento. A implementação desse problema foi através das seguintes bibliotecas do *Python*: *py2opt*, *tsp_solver* e *pymoo*, sendo realizadas 10 simulações para cada uma delas.

O programa deve atender algumas especificações, a primeira é encontrar a rota com menor deslocamento possível, atividades que exigem mais de um técnico devem ser programadas de forma que os trabalhadores atuem em conjunto e não de forma esparsa e, por fim, ter uma reprodutibilidade dos resultados, para que a operação do algoritmo transmita segurança aos utilizadores.

O planeamento gerado pelo AG anteriormente e, destacado na Tabela 5.1, utilizando os dados de entrada referidos acima, estão ilustrados na Tabela 5.16, que contém o número

de identificação da tarefa (ID), o nome do equipamento que será realizado a intervenção (Componente), a descrição completa do que deverá ser realizado (Descrição da atividade) e por fim o número de técnicos necessário para a execução do serviço (N. Tec.).

Tabela 5.7: Planeamento gerado para a simulação 4.

ID	**Componente**	**Descrição da atividade**	**N. Tec.**
2	Painel do gerador	Inspeção no painel	2
4	Painel da turbina	Inspeção no painel de comando para manutenção do exaustor do poço da turbina	2
5	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
6	Instrumento de Ensaio Eletroeletrônico	Inspeção operacional do sistema de monitorização dos geradores	2

Nota-se que todas as atividades precisam de 2 trabalhadores para serem efetuadas, portanto o algoritmo deve ser capaz de atribuir simultaneamente esse número de colaboradores ao decorrer da rota.

Simulação 4.1: algoritmo 2-opt heuristíco

A implementação desse algoritmo foi mediante da biblioteca *py2opt*, que foi justamente desenvolvida para solucionar problemas de otimização de rotas. A Tabela 5.8 apresenta os resultados obtidos, sendo eles a capacidade do programa em gerar as mesmas soluções de determinação de rotas (Eficácia da rota), seguindo o mesmo raciocínio, porém, para o cálculo do tempo de deslocamento (Eficácia do deslocamento), o tempo médio que o algoritmo levou para compilar os resultados (Tempo médio) e o seu respectivo desvio padrão do tempo (Desvio padrão da amostra).

Tabela 5.8: Resultados da simulação 4.1.

Eficácia da rota	**Eficácia do deslocamento**	**Tempo médio**	**Desvio** padrão da amostra
30%	100%	0.0160	0.002

Nota-se que o tempo de execução do programa, assim como o desvio padrão, são pequenos. O algoritmo, de fato, determinou a menor distância a ser percorrida, porém, com

uma reprodutibilidade da rota péssima de apenas 30%, sugerindo que existem caminhos distintos com a mesma distância e, como, esse pacote possui limitações não foi capaz de selecionar apenas uma rota e replicá-la, não atendendo aos objetivos do estudo.

- *Rotas* = [[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]];
- *Tempo total de deslocamento (minutos)* = [41, 41].

Simulação 4.2: algoritmo genérico ganacioso

Para essa simulação foi empregado o pacote *tsp_solver*, que também é direcionado para otimização de rotas. Foram obtidos resultados excelentes, explicitados na Tabela 5.9

Tabela 5.9: Resultados da simulação 4.2.

Eficácia da rota	**Eficácia do deslocamento**	**Tempo médio**	**Desvio** padrão da amostra
100%	100%	0.0008	0.0002

Apresentou uma reprodutibilidade 100% para determinação das rotas, sendo estas as com os menores tempos de deslocamento, além de necessitar de um tempo muito pequeno de processamento, conferindo confiabilidade ao programa. As rotas e respectivas distâncias estão descritas a seguir:

- *Rotas* = [[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]];
- *Tempo total de deslocamento (minutos)* = [41, 41].

A Figura 5.1 é uma ilustração do esquema de rotas sugeridas pelos algoritmos das Simulações 4.1 e 4.2, em que o caminho apontado pelo vetor deve ser percorrido simultaneamente pelos técnicos 1 e 2.

Simulação 4.3: algoritmo genético

Por fim, foi testado o *pymoo*, uma biblioteca mais consolidada e reconhecida do *Python*, em comparação as duas anteriores, com diversos algoritmos desenvolvidos para problemas de otimização. Os resultados obtidos encontram-se na Tabela 5.10.

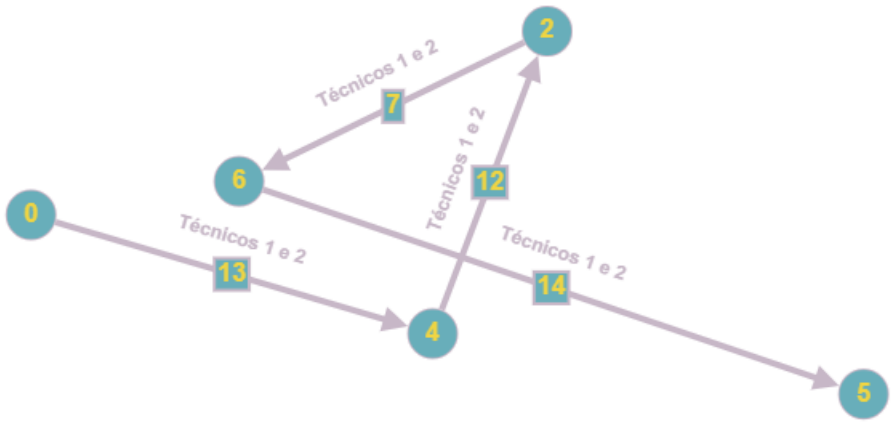


Figura 5.1: Ilustração da rota selecionada pelas Simulações 4.1 e 4.2

Tabela 5.10: Resultados da simulação 4.3.

Eficácia da rota	**Eficácia do deslocamento**	**Tempo médio**	**Desvio** padrão da amostra
100%	100%	1.7885	0.0514

Essa simulação apresentou um tempo de compilação maior que as opções anteriores, devido ao fato de que um algoritmo genético exige uma programação mais aprofundada, em especial pelo emprego de operadores genéticos. Entretanto, o algoritmo não retornou o caminho com o tempo de deslocamento mais otimizado possível, que seria de 41 minutos, como as demais.

- *Rotas* = [[0, 6, 2, 4, 5],[0, 6, 2, 4, 5]];

- *Tempo total de deslocamento (minutos)* = [45, 45].

Por fim, a Figura 5.2 é uma ilustração do esquema de rotas sugeridas pelo algoritmo da Simulação 4.3, de forma similar aponta o caminho do vetor a ser percorrido simultaneamente pelos técnicos 1 e 2.

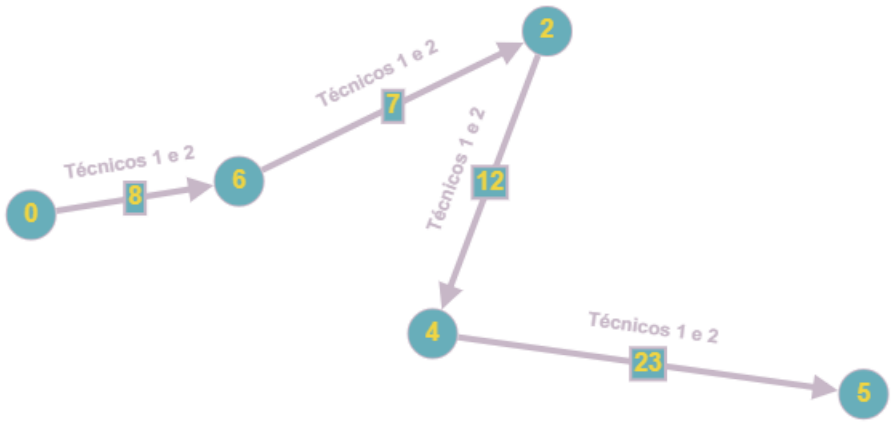


Figura 5.2: Ilustração da rota selecionada pela Simulação 4.3

5.3.2 Simulação 5: atividades com paragem da central

Para fundamentar a escolha de qual biblioteca apresenta os melhores resultados foi realizado a simulação 5 que considera que os equipamentos não estão em operação, resultando em um planeamento diferente das tarefas a serem executadas, as quais foram definidas pelo AG anteriormente, explicitado na Tabela 5.6. A Tabela 5.11, descreve em maiores detalhes o planeamento em questão:

Tabela 5.11: Planeamento gerado para a simulação 5.

ID	Componente	Descrição da atividade	N. Tec.
1	Painel do gerador	Limpeza do painel do sistema de monitorização dos geradores	2
4	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
5	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
6	Instrumento de Ensaio Eletroeletrônico	Inspeção operacional do sistema de monitorização dos geradores	2
7	Mancal escora do gerador	Inspeção na caixa de instrumentos do reservatório do mancal escora	2

Assim como no caso anterior, o algoritmo deve ser capaz de satisfazer as restrições relacionadas ao número de colaboradores exigidos na atribuição das rotas.

Simulação 5.1: algoritmo 2-opt heurístico

Afim de fundamentar a decisão de preterir o algoritmo do *py2opt* foi realizada mais uma simulação. Os resultados permanecem insatisfatórios com as restrições que o problema impõem, como demonstra a Tabela 5.12.

Tabela 5.12: Resultados da simulação 5.1.

Eficácia da rota	**Eficácia do deslocamento**	**Tempo médio**	**Desvio** padrão da amostra
30%	100%	0.021	0.002

Caso não existisse a obrigatoriedade de que os trabalhadores atuem simultaneamente nas atividades, esse pacote seria excelente, já que independente do tipo da simulação foi capaz de definir o menor caminho possível em um tempo de compilação pequeno.

- *Rotas* = [[0, 4, 7, 6, 1, 5], [0, 4, 7, 6, 1, 5]];
- *Tempo total de deslocamento (minutos)* = [51, 51].

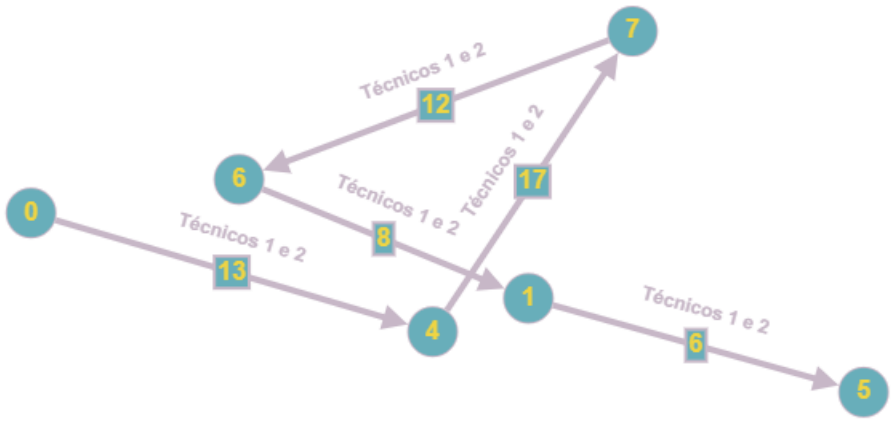


Figura 5.3: Ilustração da rota selecionada pela Simulação 5.1

Na Figura 5.3 é representado a rota ser percorrida tanto pelo técnico 1 quanto pelo técnico 2 definida pelo programa da Simualação 5.1.

Simulação 5.2: algoritmo genérico ganacioso

Dentre as simulações realizadas anteriormente, a simulação 4.2 foi o pacote que obteve melhor resultado e, em termos de replicação das respostas do algoritmo continua, a simulação 5.2 obtendo resultados satisfatórios, como aponta a Tabela 5.13

Tabela 5.13: Resultados da simulação 5.2.

Eficácia da rota	**Eficácia do deslocamento**	**Tempo médio**	**Desvio** padrão da amostra
100%	100%	0.0009	0.0005

- *Rotas* = [[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]];
- *Tempo total de deslocamento (minutos)* = [54, 54].

Entretanto para diferentes dados de entrada o programa não foi capaz de identificar o menor caminho possível, que seria de 51 minutos e não de 54 minutos. A Figura 5.4 ilustra essa rota definida pela Simulação 5.2, que é percorrida por ambos os colaboradores.

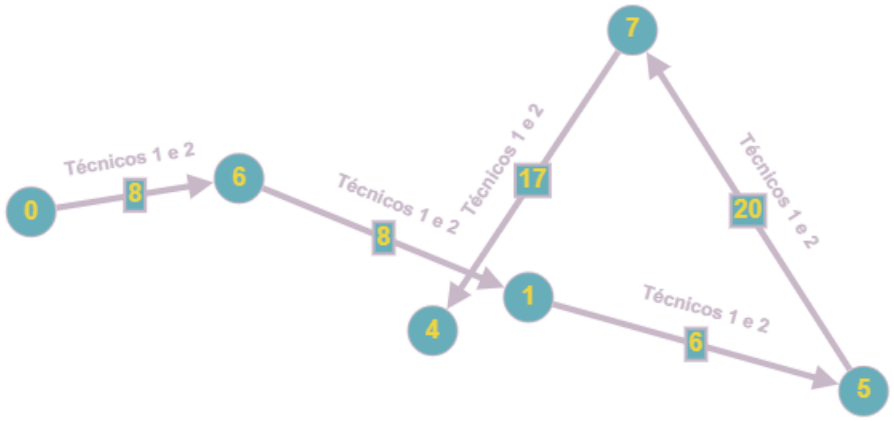


Figura 5.4: Ilustração da rota selecionada pela Simulação 5.2

Simulação 5.3: algoritmo genético

O *pymoo* permaneceu exibindo valores coerentes, ilustrados na Tabela 5.14, com uma eficácia excelente, contudo o tempo médio de compilação aumentou de cerca de 1.8 segundos

para 37.2 segundos, ainda não sendo um impeditivo para a implementação em casos reais, porém indica que quanto maior a quantidade de atividades selecionadas maior será esse tempo. Contudo, foi capaz de selecionar o menor caminho possível, diferentemente de antes.

Tabela 5.14: Resultados da simulação 5.3.

Eficácia da rota	**Eficácia do deslocamento**	**Tempo médio**	**Desvio** padrão da amostra
100%	100%	37.223	1.639

- *Rotas* = [[0, 6, 4, 7, 1, 5],[0, 6, 4, 7, 1, 5]];
- *Tempo total de deslocamento (minutos)* = [51, 51].

Por fim, a Figura 5.5 é a ilustração da rota sugerida pela Simulação 5.3, que deve ser percorrida pelos técnicos 1 e 2.

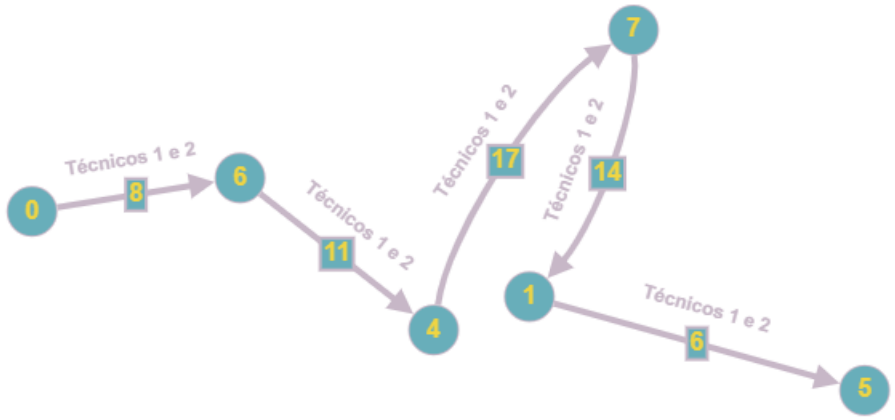


Figura 5.5: Ilustração da rota selecionada pela Simulação 5.3

Portanto, a grande diferença entre os resultados obtidos se dá na escolha da rota. Enquanto a primeira simulação, com *py2opt* apresentou resultados variados não satisfazendo as restrições do problema, as simulações do *tsp_solver* e *pymoo* obtiveram resultados razoáveis, entretanto nem sempre definindo a rota com menor caminho possível. Apesar do

pymoo exigir um tempo muito maior de execução, ainda sim é a escolha mais adequada por ser uma biblioteca confiável e capaz de suportar uma grande quantidade de dados de entrada enquanto o desempenho do *tsp_solver* é dependente da quantidade de informações sendo sua utilização limitada, assim a escolha para o algoritmo de otimização de rotas é da biblioteca *pymoo*. No Anexo B está descrito todos os testes realizados para cada um dos pacotes utilizados no estudo.

5.4 Análise dos resultados das simulações

Após a definição da melhor combinação de parâmetros do AG, os resultados da seleção e divisão de atividades rotineiras de manutenção para cada técnico estão representadas na Tabela 5.15.

Tabela 5.15: Planeamento com atividades rotineiras.

Técnico	**ID**	**Componente**	**Descrição da atividade**	**D.E.**
Técnico 1	2	Painel do gerador	Inspeção no painel	4
Técnico 1	4	Painel da turbina	Inspeção no painel de comando para manutenção do exaustor do poço da turbina	2
Técnico 1	5	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
Técnico 1	6	Instrumento de Ensaio Eletroeletrônico	Inspeção operacional do sistema de monitorização dos geradores	2
Técnico 2	2	Painel do gerador	Inspeção no painel	4
Técnico 2	4	Painel da turbina	Inspeção no painel de comando para manutenção do exaustor do poço da turbina	2
Técnico 2	5	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
Técnico 2	6	Instrumento de Ensaio Eletroeletrônico	Inspeção operacional do sistema de monitorização dos geradores	2

Em que é indicado quem é o técnico (Técnico), o ID da Máquina (ID) e seu respectivo componente (Componente), a atividade em questão a ser realizada (Descrição da atividade), e sua duração estimada (D.E.). A Figura 5.6 é apresenta a rota definida para cada

técnico, ilustrando o número de identificação de cada atividade, sua respectiva duração e a ordem de execução a ser seguida.

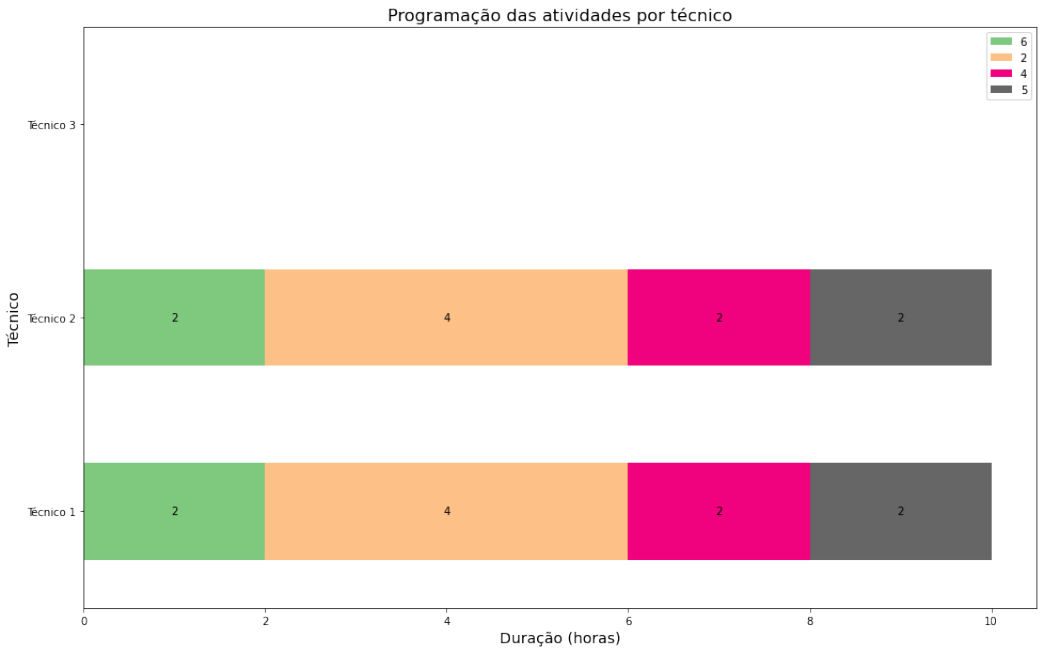


Figura 5.6: Programação das atividades rotineiras de manutenção por técnico

Nota-se que os algoritmos desenvolvidos e implementados no estudo foram não apenas capazes de cumprir com as restrições impostas pelo problema mas também otimizar os recursos disponíveis pela central hidroelétrica. Percebe-se pela Figura 5.6 que apesar de existirem três trabalhadores disponíveis, apenas dois tiveram tarefas atribuídas, devido ao fato de que as atividades, expressas na Tabela 5.15 requisitarem dois técnicos cada uma para sua execução, ou seja, houve uma otimização do corpo de trabalho.

Já para a situação em que se considera uma paragem total da unidade geradora, a atribuição das tarefas se encontram na Tabela 5.16.

É visível que foram selecionadas atividades além daquelas que necessitam que as máquinas estejam em paragem, isso representa que o programa elaborado foi capaz de otimizar a quantidade de tarefas preenchendo a duração de trabalho estipulada.

A Figura 5.7 ilustra esse planeamento gerado demonstrando novamente que o algoritmo em questão qualificou corretamente as atividades selecionadas para cada técnico.

Tabela 5.16: Planeamento com paragem da central hidroelétrica.

Técnico	**ID**	**Componente**	**Descrição da atividade**	**D.E.**
Técnico 1	1	Painel do gerador	Limpeza do painel do sistema de monitorização dos geradores	2
Técnico 1	4	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
Técnico 1	5	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
Técnico 1	6	Instrumento de Ensaio Eletroeletrônico	Inspeção operacional do sistema de monitorização dos geradores	2
Técnico 1	7	Mancal escora do gerador	Inspeção na caixa de instrumentos do reservatório do mancal escora	2
Técnico 2	1	Painel do gerador	Limpeza do painel do sistema de monitorização dos geradores	2
Técnico 2	4	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
Técnico 2	5	Motor da bomba do regulador de velocidade	Lubrificação no motor da bomba	2
Técnico 2	6	Instrumento de Ensaio Eletroeletrônico	Inspeção operacional do sistema de monitorização dos geradores	2
Técnico 2	7	Mancal escora do gerador	Inspeção na caixa de instrumentos do reservatório do mancal escora	2

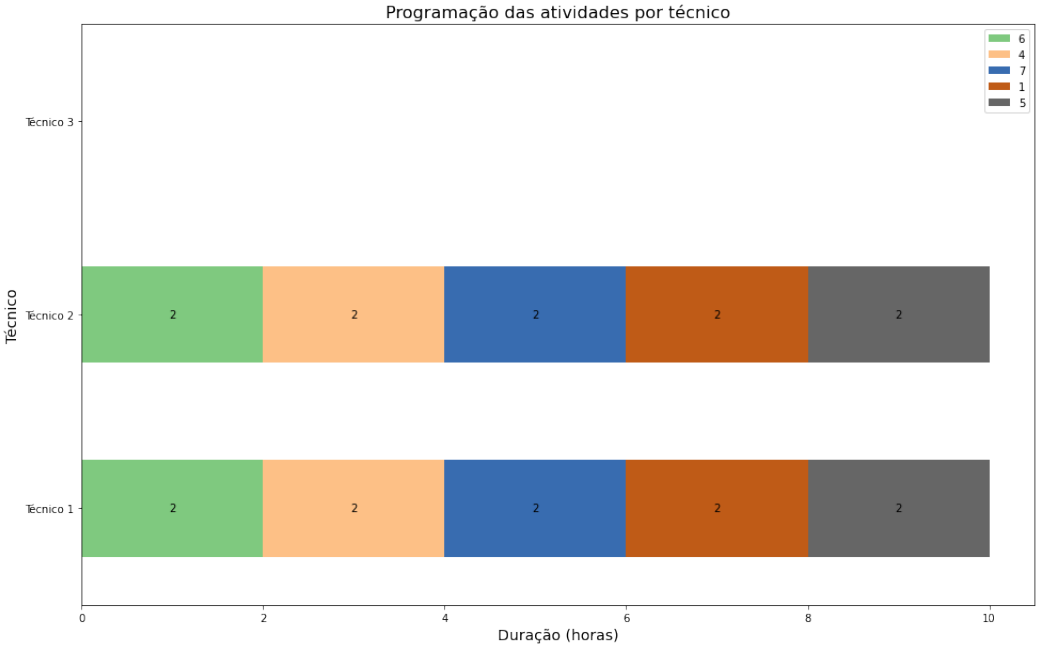


Figura 5.7: Programação das atividades de manutenção por técnico com paragem da central

Capítulo 6

Considerações finais

As políticas de manutenção são essenciais para o funcionamento eficiente de uma central hidroelétrica, para que esta possa atingir sua função principal, que é a geração de energia. Um planeamento adequado desse setor melhora a confiabilidade e a disponibilidade de equipamentos que, por sua vez, reduzem perdas económicas e elevam a produção da planta.

Um sistema hidroelétrico é uma instalação complexa, contendo uma grande quantidade de máquinas, cada uma com suas características e especificidades, tornando-se um desafio elaborar um planeamento assertivo.

Foi objetivo deste estudo desenvolver um conjunto de contribuições significativas para a otimização do problema de planeamento do setor de manutenção, através da implementação de modelos de otimização. Estes modelos devem ser capazes de desenvolver uma metodologia que permita selecionar atividades de manutenção e atribui-las para os técnicos em rotas otimizadas.

Para a seleção de tarefas foi desenvolvido um algoritmo genético (AG) implementando a biblioteca *pyeasyga*. Foram realizadas simulações para analisar a melhor combinação de parâmetros do AG, pode-se concluir que a escolha adequada destes influenciam no desempenho do programa permitindo a obtenção de resultados melhores com uma maximização da quantidade de atividades escolhidas.

Em relação ao problema de otimização de rotas, foi utilizado três bibliotecas do *Python*,

sendo eles o *py2opt*, *tsp_solver* e *pymoo*, cada um com uma abordagem diferente na resolução. Foi definido o *pymoo* como a melhor solução, pelo fato de ser um pacote consolidado na comunidade *Python* e capaz de suportar uma grande quantidade de dados, mas apresenta oportunidades de melhorias em relação ao tempo de compilação do programa e da otimização da função objetivo para uma definição mais adequada das rotas.

De uma forma geral, conclui-se que as modelações dos problemas são capazes de aproximar-se à realidade do planeamento da manutenção de uma central hidroelétrica, ou seja, o programa responde de forma realista a estruturação de rotas otimizadas de manutenção. A metodologia empregada, também, permite lidar com as restrições técnicas que caracterizam este problema, adaptando-se a diferentes cenários que esse setor enfrenta para o planeamento das intervenções de manutenção.

De acordo com os resultados obtidos, é possível afirmar que a implementação de um programa baseado em Algoritmos Genéticos é adequado para a otimização de problemas relacionados ao planeamento de atividades da manutenção de centrais hidroelétricas, atendendo aos objetivos do estudo.

Para trabalhos futuros sugere-se:

- Neste trabalho uma das simplificações utilizadas, foi a desconsideração de diferentes especialidades dos técnicos, como por exemplo técnicos de mecânica e elétrica; seria de grande interesse desenvolver um programa capaz de realizar essa diferenciação no planeamento de atividades;

- Explorar outros tipos de algoritmos como o *Particle Swarm Optimization*, e realizar uma comparação dos resultados com o algoritmo genético;

- Fazer diferentes funções objetivo do *pymoo* para uma obtenção mais precisa das rotas;

- Obter informações sobre trocas de componentes de cada atividade e relacionar com disponibilidade de inventário;

- Aplicar algum algoritmo de *machine learning* para a divisão de atividades para cada técnico.

Bibliografia

[1] Anis Allal, M'Hammed Sahnoun, Réda Adjoudj, Sidi Mohamed Benslimane, and Merouane Mazar. Multi-agent based simulation-optimization of maintenance routing in offshore wind farms. *Computers Industrial Engineering*, 157:107342, 2021.

[2] Sara Antomarioni, Marjorie Maria Bellinello, Maurizio Bevilacqua, Filippo Emanuele Ciarapica, Renan Favarão da Silva, and Gilberto Francisco Martha de Souza. A data-driven approach to extend failure analysis: A framework development and a case study on a hydroelectric power plant. *Energies*, 13(23), 2020.

[3] Pedram Ataee. How to solve the traveling salesman problem with the 2-opt algorithm, v1.3.6. `https://pypi.org/project/py2opt/`, 2021. Last accessed on: 2022.11.08.

[4] Beatriz Flamia Azevedo. Study of genetic algorithm for optimization problems. Instituto Politécnico de Bragança, 2020.

[5] Ernnie Illyani Basri, Izatul Hamimi Abdul Razak, Hasnida Ab-Samat, and Shahrul Kamaruddin. Preventive maintenance (pm) planning: a review. *Journal of Quality in Maintenance Engineering*, 23(2), 2017.

[6] Marjorie M. Bellinello, S. Antomarioni, G. F. M. Souza, M. Bevilacqua, and F. E. Ciarapica. Entropy-maut integrated approach supported by fuzzy k-means: a robust tool for determining critical components for maintenance monitoring and a case study of kaplan hydro generator unit. *Production*, 32(e20210066), 2022.

[7] Marjorie M. Bellinello, Miguel A. C. Michalski, Arthur H. A. Melani, Adherbal Caminada Netto, Carlos A. Murad, and Gilberto F. M. Souza. Pal-vmea: A novel method for enhancing decision-making consistency in maintenance management. *Applied Sciences*, 10(22), 2020.

[8] Julian Blank. pymoo: Multi-objective optimization in python, v0.6.0. `https://pymoo.org/`, 2022. Last accessed on: 2022.10.30.

[9] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[10] Anabela Botelho, Paula Ferreira, Fátima Lima, Lígia M. Costa Pinto, and Sara Sousa. Assessment of the environmental impacts associated with hydropower. *Renewable and Sustainable Energy Reviews*, 70:896–904, 2017.

[11] Marcello Braglia, Davide Castellano, and Mosè Gallo. A novel operational approach to equipment maintenance: Tpm and rcm jointly at work. *Journal of Quality in Maintenance Engineering*, 2019.

[12] Paul Breeze. *Power generation technologies*. Newnes, Oxford, Oxfordshire, Reino Unido, 3st edition, 2019.

[13] Merve Bulut and Evrencan Özcan. A new approach to determine maintenance periods of the most critical hydroelectric power plant equipment. *Reliability Engineering System Safety*, 205:107238, 2021.

[14] Mikel Canizo, Enrique Onieva, Angel Conde, Santiago Charramendieta, and Salvador Trujillo. Real-time predictive maintenance for wind turbines using big data frameworks. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 70–77, 2017.

[15] Silvia Carpitella, Ilyas Mzougui, Julio Benítez, Fortunato Carpitella, Antonella Certa, Joaquín Izquierdo, and Marco La Cascia. A risk evaluation framework for

the best maintenance strategy: The case of a marine salt manufacture firm. *Reliability Engineering System Safety*, 205:107265, 2021.

[16] Thomas Caswell, Antony Lee, Michael Droettboom, and et al. matplotlib/matplotlib: Rel: v3.6.2. `https://doi.org/10.5281/zenodo.592536.svg`, November 2022.

[17] Adolfo Crespo Marquez, Juan Francisco Gomez Fernandez, Pablo Martínez-Galán Fernández, and Antonio Guillen Lopez. Maintenance management through intelligent asset management platforms (iamp). emerging factors, key impact areas and data models. *Energies*, 13(15), 2020.

[18] Instituto Português da Qualidade. Norma portuguesa np en 13306 2007: terminologia da manutenção. Caparica: IPQ, 2007. Last accessed on 2022.11.14.

[19] Ministério de Minas e Energia. Brazilian energy balance 2021, 2021. Avaliable online at: `https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/balanco-energetico-nacional-2021`. Last accessed on 2022.02.27.

[20] Xueshi Dong, Hong Zhang, Min Xu, and Fanfan Shen. Hybrid genetic algorithm with variable neighborhood search for multi-scale multiple bottleneck traveling salesmen problem. *Future Generation Computer Systems*, 114:229–242, 2021.

[21] Python Software Foundation. Python package index - pypi. `https://pypi.org/`. Last accessed on 2022.11.27.

[22] Samira Gholizadeh. Top Popular Python Libraries in Research. *Journal of Robotics and Automation Research*, 3(2):142–145, 2022.

[23] Jesús Gil, Javier Martinez Torres, and Rubén González-Crespo. The application of artificial intelligence in project management research: A review. (6):54–66, 2021.

[24] Ramesh Gulati. *Maintenance and Reliability Best Practices*. Industrial Press, Inc., Nova York, Nova York, ESTADOS UNIDOS, 3st edition, 2020.

[25] Nurul Fadly Habidin, Suzaituladwini Hashim, Nursyazwani Mohd Fuzi, and Mad Ithnin Salleh. Total productive maintenance, kaizen event, and performance. *International Journal of Quality & Reliability Management*, 35(9):1853–186, 2018.

[26] Putri Mutira Hariyadi, Phong Thanh Nguyen, Iswanto Iswanto, and Dadang Sudrajat. Traveling salesman problem solution using genetic algorithm. *Journal of Critical Reviews*, 7(1):56–61, 2020.

[27] Charles Harris, Kenneth Jarrod Millman, Stéfan van der Walt, and et al. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[28] Stefan Hougardy, Fabian Zaiser, and Xianghui Zhong. The approximation ratio of the 2-opt heuristic for the metric traveling salesman problem. *Operations Research Letters*, 48(4):401–404, 2020.

[29] Jing Huang, Qing Chang, and Jorge Arinez. Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications*, 160:113701, 2020.

[30] Itaipu. Unidades geradoras. , 2010. Avaliable online at: `https://www.itaipu.gov.br/energia/unidades-geradoras`. Last accessed on 2022.03.04.

[31] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

[32] Farid Ullah Khan, Wahad ur Rahman, and Muhammad Masood Ahmad. Modeling, simulation and fabrication of micro kaplan turbine. *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH*, 10(6), 2021.

[33] Myungseob Kim, Nikola Marković, and Eungcheol Kim. A vertical railroad alignment design with construction and operating costs. *Journal of Transportation Engineering, Part A: Systems*, 145(10):04019043, 2019.

[34] Sharon Klein and Emma Fox. A review of small hydropower performance and cost. *Renewable and Sustainable Energy Reviews*, 169:112898, 2022.

[35] Oliver Kramer. Genetic algorithms. In *Genetic algorithm essentials*, pages 11–19. Springer, 2017.

[36] Krishna Kumar and R.P. Saini. A review on operation and maintenance of hydropower plants. *Sustainable Energy Technologies and Assessments*, 49:101704, 2022.

[37] Krishna Kumar, Ajay Kumar Singh, and Ravindra Pratap Singh. Synchronous condensing mode operation of hydropower plants. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 486–488. IEEE, 2020.

[38] Krishna Kumar, Ajay Kumar Singh, and Ravindra Pratap Singh. Synchronous condensing mode operation of hydropower plants. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 486–488, 2020.

[39] Nahyun Kwon, Kwonsik Song, Yonghan Ahn, Moonsun Park, and Youjin Jang. Maintenance cost prediction for aging residential buildings based on case-based reasoning and genetic algorithm. *Journal of Building Engineering*, 28:101006, 2020.

[40] Guillaume Lemaitre. scikit-learn/scikit-learn: scikit-learn 1.1.3. `https://doi.org/10.5281/zenodo.7254371`, October 2022.

[41] Ruopu Li, Ehsan Arzaghi, Rouzbeh Abbassi, Diyi Chen, Chunhao Li, Huanhuan Li, and Beibei Xu. Dynamic maintenance planning of a hydro-turbine in operational life cycle. *Reliability Engineering System Safety*, 204:107129, 2020.

[42] Camilla Lundgren, Jon Bokrantz, and Anders Skoogh. A strategy development process for smart maintenance implementation. *Journal of Manufacturing Technology Management*, 32(9):142–166, 2021.

[43] Camilla Lundgren, Anders Skoogh, and Jon Bokrantz. Quantifying the effects of maintenance – a literature review of maintenance models. *Procedia CIRP*, 72:1305–1310, 2018. 51st CIRP Conference on Manufacturing Systems.

[44] Eduyn López-Santana, Raha Akhavan-Tabatabaei, Laurence Dieulle, Nacima Labadie, and Andrés L. Medaglia. On the combined maintenance and routing optimization problem. *Reliability Engineering System Safety*, 145:199–214, 2016.

[45] Francisco Javier Martinez-Monseco. An approach to a maintenance plan for a turbine of hydroelectric power plant. optimisation based in rcm and fmeca analysis. *Journal of Applied Research in Technology & Engineering*, 2(1):39–50, 2021.

[46] Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* O'Reilly Media Inc., Sebastopol, Califórnia, Estados Unidos, 2ed. edition, 2017.

[47] Peter Muchiri, Liliane Pintelon, Ludo Gelders, and Harry Martin. Development of maintenance function performance measurement framework and indicators. *International Journal of Production Economics*, 131(1):295–302, 2011. Innsbruck 2008.

[48] Abhinav Nagpal and Goldie Gabrani. Python for data analytics, scientific and technical applications. In *2019 Amity international conference on artificial intelligence (AICAI)*, pages 140–145. IEEE, 2019.

[49] Miguel Angel Navas, Carlos Sancho, and Jose Carpio. Disruptive maintenance engineering 4.0. *International Journal of Quality & Reliability Management*, 37(6/7):853–871, 2020.

[50] Adherbal Caminada Netto, Arthur Henrique de Andrade Melani, Carlos Alberto Murad, Miguel Angelo de Carvalho Michalski, Gilberto Francisco Martha de Souza, and Silvio Ikuyo Nabeta. A novel approach to defining maintenance significant items: A hydro generator case study. *Energies*, 13(23), 2020.

[51] Ho Si Hung Nguyen, Phuc Do, Hai-Canh Vu, and Benoit Iung. Dynamic maintenance grouping and routing for geographically dispersed production systems. *Reliability Engineering System Safety*, 185:392–404, 2019.

[52] Filscha Nurprihatin, Meilily Angely, and Hendy Tannady. Total productive maintenance policy to increase effectiveness and maintenance performance using overall equipment effectiveness. *Journal of applied research on industrial engineering*, 6(3):184–199, 2019.

[53] Evrencan Özcan, Gür Şeyda, and Eren Tamer. A hybrid model to optimize the maintenance policies in the hydroelectric power plants. *Politeknik Dergisi*, (1):75–86, 2020.

[54] The pandas development team. pandas-dev/pandas: Pandas, v1.5.2. `https://doi.org/10.5281/zenodo.3509134`, February 2020.

[55] Martin Pech, Jaroslav Vrchota, and Jiří Bednář. Predictive maintenance and intelligent sensors in smart factory: Review. *Sensors*, 21(4), 2021.

[56] Geraldo Magela Pereira. *Projetos de Usinas Hidrelétricas*. Oficina de Textos, São Paulo, São Paulo, Brasil, 1ed. edition, 2015.

[57] Python. O tutorial python — documentação python 3.10.5, 2021. Avaliable online at: `https://docs.python.org/pt-br/3/tutorial/`. Last accessed on 2022.06.10.

[58] Emanuele Quaranta and Peter Davies. Emerging and innovative materials for hydropower engineering applications: Turbines, bearings, sealing, dams and waterways, and ocean power. *Engineering*, 8:148–158, 2022.

[59] Emanuele Quaranta and Chirag Trivedi. The state-of-art of design and research for pelton turbine casing, weight estimation, counterpressure operation and scientific challenges. *Heliyon*, 7(12):e08527, 2021.

[60] Lineu Belico dos Reis. *Geracão de Energia Elétrica*. Manole Ltda., Barueri, São Paulo, Brasil, 3ed. edition, 2017.

[61] Ayodeji Remi-Omosowon. pyeasyga, 2015. Avaliable online at: `https://pyeasyga.readthedocs.io/en/latest/#`. Last accessed on 2022.07.24.

[62] Abdellah Rezoug, Mohamed Bader-El-Den, and Dalila Boughaci. Guided genetic algorithm for the multidimensional knapsack problem. *Memetic Computing*, 10(1):29–42, 2018.

[63] Edson Ruschel, Eduardo Alves Portela Santos, and Eduardo de Freitas Rocha Loures. Industrial maintenance decision-making: A systematic literature review. *Journal of Manufacturing Systems*, 45:180–194, 2017.

[64] Farrukh Shaazizov, Abdulla Badalov, Alisher Ergashev, and Diyor Shukurov. Studies of rational methods of water selection in water intake areas of hydroelectric power plants. In *E3S Web of Conferences*, volume 97, page 05041. EDP Sciences, 2019.

[65] Dalia Abdulkareem Shafiq, Mohsen Marjani, Riyaz Ahamed Ariyaluran Habeeb, and David Asirvatham. Student retention using educational data mining and predictive analytics: A systematic literature review. *IEEE Access*, 10:72480–72503, 2022.

[66] Sonia Sharma and Shantu Kar. Risk management and analysis in hydro-electric projects in india. *SSRG International Journal of Civil Engineering*, 5:1–7, 2018.

[67] Dmitry Shintyakov. Greedy, suboptimal solver for the travelling salesman problem, v0.4.1. `https://pypi.org/project/tsp-solver2/`, July 2020. Last accessed on: 2022.11.16.

[68] Igor Stančin and Alan Jović. An overview and comparison of free python libraries for data mining and big data analysis. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 977–982. IEEE, 2019.

[69] Chengguo Su, Peilin Wang, Wenlin Yuan, Chuntian Cheng, Taiheng Zhang, Denghua Yan, and Zening Wu. An milp based optimization model for reservoir flood control operation considering spillway gate scheduling. *Journal of Hydrology*, 613:128483, 2022.

[70] Kong Fah Tee and Ejiroghene Ekpiwhre. Reliability-based preventive maintenance strategies of road junction systems. *International Journal of Quality & Reliability Management*, 36(5):752–781, 2019.

[71] Rafia Nishat Toma, Alexander E. Prosvirin, and Jong-Myon Kim. Bearing fault diagnosis of induction motors using a genetic algorithm and machine learning classifiers. *Sensors*, 20(7), 2020.

[72] Guilherme Luz Tortorella, Flavio S. Fogliatto, Paulo A. Cauchick-Miguel, Sherah Kurnia, and Daniel Jurburg. Integration of industry 4.0 technologies into total productive maintenance practices. *International Journal of Production Economics*, 240:108224, 2021.

[73] Raymon van Dinter, Bedir Tekinerdogan, and Cagatay Catal. Predictive maintenance using digital twins: A systematic literature review. *Information and Software Technology*, 151:107008, 2022.

[74] Pauli Virtanen, Ralf Gommers, Travis Oliphant, Matt Haberland, and et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[75] Voith. Kaplan turbines. Avaliable online at: `https://voith.com/at-en/turbines-generators/turbines/kaplan-turbines.html`. Last accessed on 2022.03.04.

[76] Pascal Vrignat, Frédéric Kratz, and Manuel Avila. Sustainable manufacturing, maintenance policies, prognostics and health management: A literature review. *Reliability Engineering System Safety*, 218:108140, 2022.

[77] Weifeng Wang, Bing Lou, Xiong Li, Xizhong Lou, Ning Jin, and Ke Yan. Intelligent maintenance frameworks of large-scale grid using genetic algorithm and k-mediods clustering methods. *World Wide Web*, 23(2):1177–1195, 2020.

[78] Halarius G. Xenos. *Gerenciando a manutenção produtiva: o caminho para eliminar falhas nos equipamentos e aumentar a produtividade*. Falconi, Belo Horizonte, Minas Gerais, BRASIL, 2st edition, 2014.

[79] Li Da Xu, Eric L. Xu, and Ling Li. Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018.

[80] Ming Zhang, David Valentín, Carme Valero, Mònica Egusquiza, and Eduard Egusquiza. Failure investigation of a kaplan turbine blade. *Engineering Failure Analysis*, 97:690–700, 2019.

Apêndice A

Simulações dos parâmetros do AG

Tabela A.1: Combinações de parâmetros do AG

Número	**População**	**Geração**	***Crossover***	**Mutação**
Combinação 1	100	30	0,8	0,05
Combinação 2	100	30	0,8	0,1
Combinação 3	100	30	0,9	0,05
Combinação 4	100	30	0,9	0,1
Combinação 5	100	60	0,8	0,05
Combinação 6	100	60	0,8	0,1
Combinação 7	100	60	0,9	0,05
Combinação 8	100	60	0,9	0,1
Combinação 9	200	60	0,8	0,05
Combinação 10	200	60	0,8	0,1
Combinação 11	200	60	0,9	0,05
Combinação 12	200	60	0,9	0,1
Combinação 13	200	30	0,8	0,05
Combinação 14	200	30	0,8	0,1
Combinação 15	200	30	0,9	0,05
Combinação 16	200	30	0,9	0,1

Tabela A.2: Simulações dos parâmetros do AG sem elitismo

Reprodução sem elitismo				
Parâmetros	**Execução**	**Pontuação**	**Atividades selecionadas**	**Tempo de execução**
Combinação 1	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0681
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0640
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0678
	4	3,7119	[1, 1, 0, 1, 0, 1, 0]	0,0712
	5	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,0862
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0830
	7	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,0836
	8	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,0583
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0667
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0562
Combinação 2	1	3,0216	[1, 0, 1, 0, 1, 0, 0]	0,0600
	2	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,0945
	3	2,4739	[1, 0, 0, 1, 0, 1, 0]	0,1035
	4	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,0636
	5	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,0626
	6	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,0590
	7	4,3582	[0, 0, 1, 1, 1, 0, 0]	0,0636
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0800
	9	4,3582	[0, 0, 1, 1, 1, 0, 0]	0,0917
	10	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,0600
Combinação 3	1	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,0768
	2	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,0638
	3	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,0751
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0667

	5	4,3582	[0, 0, 1, 1, 1, 0, 0]	0,0708
	6	5,3969	[1, 1, 0, 0, 1, 1, 0]	0,0740
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0591
	8	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,0626
	9	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,0592
	10	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1077
Combinação 4	1	4,3582	[0, 0, 1, 1, 1, 0, 0]	0,0580
	2	5,3969	[0, 1, 0, 0, 1, 1, 0]	0,0770
	3	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,0625
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0724
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0679
	6	5,3969	[1, 1, 0, 0, 1, 1, 0]	0,0618
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0669
	8	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,0615
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0573
	10	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,0692
Combinação 5	1	5,3969	[1, 1, 0, 0, 1, 1, 0]	0,1218
	2	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1078
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1042
	4	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1078
	5	1,2380	[0, 1, 0, 0, 0, 0, 0]	0,1054
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1035
	7	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1062
	8	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1059
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1059
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1045
Combinação 6	1	5,3969	[0, 1, 0, 0, 1, 1, 0]	0,1708
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1218

	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1347
	4	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1467
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1294
	6	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1296
	7	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1125
	8	5,3969	[1, 1, 0, 0, 1, 1, 0]	0,1141
	9	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1171
	10	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1126
Combinação 7	1	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1539
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1294
	3	4,1589	[0, 0, 0, 0, 1, 1, 0]	0,1161
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1208
	5	4,3582	[1, 0, 0, 1, 1, 0, 0]	0,1361
	6	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1412
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1274
	8	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1579
	9	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1172
	10	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1144
Combinação 8	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1343
	2	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1689
	3	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1537
	4	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1126
	5	2,5746	[0, 1, 0, 1, 0, 0, 0]	0,1248
	6	4,3582	[1, 0, 0, 1, 1, 0, 0]	0,1292
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1275
	8	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1222
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1235
	10	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1227

Combinação 9	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1226
	2	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1359
	3	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1329
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1148
	5	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1231
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1109
	7	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1278
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1308
	9	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1353
	10	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1294
Combinação 10	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1099
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1139
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1072
	4	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1296
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1084
	6	5,3969	[1, 1, 0, 0, 1, 1, 0]	0,1080
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1087
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1098
	9	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1065
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1220
Combinação 11	1	5,3969	[0, 1, 0, 0, 1, 1, 0]	0,1076
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1079
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1029
	4	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,1054
	5	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1049
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1083
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1110
	8	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1098

	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1075
	10	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1066
Combinação 12	1	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1210
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1038
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1033
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1050
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1189
	6	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,1103
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1069
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1098
	9	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,1044
	10	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,1065
Combinação 13	1	4,3582	[0, 0, 1, 1, 1, 0, 0]	0,2280
	2	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,2109
	3	5,3969	[0, 1, 0, 0, 1, 1, 0]	0,2101
	4	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,2091
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2099
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2111
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2251
	8	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,2249
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2320
	10	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,2098
Combinação 14	1	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,2121
	2	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,2079
	3	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,2066
	4	5,3969	[0, 1, 0, 0, 1, 1, 0]	0,2100
	5	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,2043
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2110

	7	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,2079
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2030
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2089
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2044
Combinação 15	1	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,2337
	2	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,2173
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2105
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2118
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2148
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2100
	7	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,2064
	8	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,2090
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2075
	10	5,5962	[0, 1, 0, 1, 1, 0, 0]	0,2082
Combinação 16	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2269
	2	5,4955	[1, 0, 0, 1, 1, 1, 0]	0,2134
	3	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,2148
	4	5,3969	[1, 1, 0, 0, 1, 1, 0]	0,2129
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2140
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2122
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2106
	8	5,4955	[0, 0, 0, 1, 1, 1, 0]	0,2115
	9	5,5962	[1, 1, 0, 1, 1, 0, 0]	0,2148
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2170

Tabela A.3: Simulações dos parâmetros do AG com elitismo

Reprodução com elitismo				
Parâmetros	**Execução**	**Pontuação**	**Atividades selecionadas**	**Tempo de execução**
Combinação 1	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0546
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0540
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0520
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0535
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0542
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0615
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0779
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0641
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0529
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0569
Combinação 2	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0555
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0534
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0575
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0590
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0537
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0531
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0534
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0527
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0548
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0538
Combinação 3	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0688
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0543
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0580
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0548

	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0562
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0539
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0617
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0524
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0556
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0534
Combinação 4	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0779
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0591
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0534
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0534
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0507
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0553
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0519
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0589
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0530
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,0541
Combinação 5	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1114
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1061
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1103
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1059
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1088
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1471
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1335
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1062
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1089
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1235
Combinação 6	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1349
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1342

	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1061
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1412
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1496
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1166
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1085
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1707
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1286
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1078
Combinação 7	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1322
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1077
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1103
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1092
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1199
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1160
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1785
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1085
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1093
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1071
Combinação 8	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1563
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1237
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1096
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1074
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1091
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1079
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1431
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1305
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1115
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1137

Combinação 9	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1460
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1059
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1061
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1096
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1067
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1049
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1067
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1070
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1061
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1400
Combinação 10	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1184
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1174
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1438
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1033
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1049
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1070
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1769
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1062
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1127
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1048
Combinação 11	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1659
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1093
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1565
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1364
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1029
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1082
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1747
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1163

	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1052
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1076
Combinação 12	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1560
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1055
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1076
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1757
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1071
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1062
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1076
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1140
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1641
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,1069
Combinação 13	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2624
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2190
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2376
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2547
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2185
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2223
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2752
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2545
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2201
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2351
Combinação 14	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2749
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2271
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2160
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2265
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2554
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2266

	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2236
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2439
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2559
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2342
Combinação 15	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2254
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2114
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2180
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2877
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2676
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2253
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2724
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2144
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2124
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2738
Combinação 16	1	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2793
	2	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2188
	3	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2878
	4	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2120
	5	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2658
	6	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2404
	7	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2127
	8	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2643
	9	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2528
	10	6,7335	[0, 1, 0, 1, 1, 1, 0]	0,2121

Apêndice B

Simulações dos algoritmos para otimização de rotas

Tabela B.1: Resultados das simulações para atividades rotineiras de manutenção - py2opt

py2opt			
Execução	**Rota**	**Deslocamento (minutos)**	**Tempo de execução**
1	[[0, 4, 2, 6, 5], [0, 2, 4, 6, 5]]	[41, 41]	0.0158
2	[[0, 2, 4, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.0152
3	[[0, 2, 4, 6, 5], [0, 2, 4, 6, 5]]	[41, 41]	0.0172
4	[[0, 4, 2, 6, 5], [0, 2, 4, 6, 5]]	[41, 41]	0.0196
5	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.0142
6	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.0130
7	[[0, 2, 4, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.0150
8	[[0, 2, 4, 6, 5], [0, 2, 4, 6, 5]]	[41, 41]	0.0157
9	[[0, 4, 2, 6, 5], [0, 2, 4, 6, 5]]	[41, 41]	0.0168
10	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.0179

Tabela B.2: Resultados das simulações para atividades rotineiras de manutenção - tsp_solver

tsp_solver			
Execução	**Rota**	**Deslocamento (minutos)**	**Tempo de execução**
1	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00036
2	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00055
3	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00102
4	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00103
5	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00099
6	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00098
7	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00079
8	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00092
9	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00098
10	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[41, 41]	0.00099

Tabela B.3: Resultados das simulações para atividades rotineiras de manutenção - pymoo

pymoo			
Execução	**Rota**	**Deslocamento (minutos)**	**Tempo de execução**
1	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.8032
2	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.7481
3	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.8102
4	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.7069
5	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.7635
6	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.7508
7	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.7700
8	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.8076
9	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.8870
10	[[0, 6, 2, 4, 5], [0, 6, 2, 4, 5]]	[45, 45]	1.8375

Tabela B.4: Resultados das simulações para atividades com paragem da central - py2opt

py2opt			
Execução	**Rota**	**Deslocamento (minutos)**	**Tempo de execução**
1	[[0, 4, 7, 6, 1, 5], [0, 6, 7, 4, 1, 5]]	[51, 51]	0.0223
2	[[0, 4, 7, 6, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	0.0205
3	[[0, 4, 7, 6, 1, 5], [0, 7, 6, 4, 1, 5]]	[51, 51]	0.0212
4	[[0, 4, 7, 6, 1, 5], [0, 4, 7, 6, 1, 5]]	[51, 51]	0.0209
5	[[0, 4, 7, 6, 1, 5], [0, 4, 6, 7, 1, 5]]	[51, 51]	0.0187
6	[[0, 6, 4, 7, 1, 5], [0, 7, 6, 4, 1, 5]]	[51, 51]	0.0240
7	[[0, 6, 7, 4, 1, 5], [0, 6, 7, 4, 1, 5]]	[51, 51]	0.0205
8	[[0, 4, 6, 7, 1, 5], [0, 6, 7, 4, 1, 5]]	[51, 51]	0.0200
9	[[0, 4, 7, 6, 1, 5], [0, 4, 7, 6, 1, 5]]	[51, 51]	0.0242
10	[[0, 6, 4, 7, 1, 5], [0, 4, 7, 6, 1, 5]]	[51, 51]	0.0158

Tabela B.5: Resultados das simulações para atividades com paragem da central - tsp_solver

tsp_solver			
Execução	**Rota**	**Deslocamento (minutos)**	**Tempo de execução**
1	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00097
2	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00098
3	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00099
4	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00056
5	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00052
6	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00213
7	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00099
8	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00099
9	[[0, 4, 2, 6, 5], [0, 4, 2, 6, 5]]	[54, 54]	0.00056
10	[[0, 6, 1, 5, 7, 4], [0, 6, 1, 5, 7, 4]]	[54, 54]	0.00057

Tabela B.6: Resultados das simulações para atividades com paragem da central - pymoo

pymoo			
Execução	**Rota**	**Deslocamento (minutos)**	**Tempo de execução**
1	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	36.8684
2	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	39.7566
3	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	37.0026
4	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	37.1536
5	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	37.8188
6	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	37.3168
7	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	39.7217
8	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	35.9727
9	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	34.2893
10	[[0, 6, 4, 7, 1, 5], [0, 6, 4, 7, 1, 5]]	[51, 51]	36.3260