

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS - CÂMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GIOVANI CANDIDO

**UMA APLICAÇÃO WEB PARA ANÁLISE COMPARATIVA DE
META-HEURÍSTICAS DE OTIMIZAÇÃO**

BAURU

Janeiro/2023

GIOVANI CANDIDO

UMA APLICAÇÃO WEB PARA ANÁLISE COMPARATIVA DE META-HEURÍSTICAS DE OTIMIZAÇÃO

Trabalho de Conclusão do Curso de Bacharelado
em Ciência da Computação da Universidade
Estadual Paulista “Júlio de Mesquita Filho”,
Faculdade de Ciências, Câmpus de Bauru.
Orientador: Prof. Dr. João Paulo Papa
Coorientador: Dr. Leandro Aparecido Passos
Júnior

BAURU
Janeiro/2023

C217a Candido, Giovani
Uma Aplicação Web para Análise Comparativa de Meta-heurísticas de Otimização / Giovani Candido. -- Bauru, 2023
70 p. : il.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: João Paulo Papa
Coorientador: Leandro Aparecido Passos Júnior

1. Aplicação Web. 2. Análise Comparativa. 3. Meta-heurísticas. 4. Funções de Teste. 5. Otimização Matemática. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Giovani Candido

Uma Aplicação Web para Análise Comparativa de Meta-heurísticas de Otimização

Trabalho de Conclusão do Curso de Bacharelado
em Ciência da Computação da Universidade
Estadual Paulista "Júlio de Mesquita Filho",
Faculdade de Ciências, Câmpus de Bauru.

Banca Examinadora

Prof. Dr. João Paulo Papa

Orientador

Departamento de Computação

Faculdade de Ciências

Universidade Estadual Paulista "Júlio de
Mesquita Filho"

**Profa. Dra. Simone das Graças
Domingues Prado**

Departamento de Computação

Faculdade de Ciências

Universidade Estadual Paulista "Júlio de
Mesquita Filho"

**Prof. Dr. Kelton Augusto Pontara da
Costa**

Departamento de Computação

Faculdade de Ciências

Universidade Estadual Paulista "Júlio de
Mesquita Filho"

Bauru, _____ de janeiro de 2023.

Dedico este trabalho a minha família, a minha parceira, aos meus amigos e a todos que estiveram presentes durante esta longa jornada.

Agradecimentos

Agradeço, em primeiro lugar, a Deus, por me agraciar com inúmeras oportunidades e sempre revigorar minha força de vontade para que eu possa lidar com os eventuais desafios.

Agradeço, em segundo lugar, a minha família, em especial aos meus pais, Elaine Cristina Rufino Candido e Luiz Fabiano Candido, e aos meus avós, Luiz Carlos Rufino e Conceição Aparecida Segantin Rufino, pelo carinho, sustento, motivação e apoio durante minha jornada.

Agradeço também a minha companheira, Laiz Fernanda Alves Ferreira, por sua presença, carinho, compreensão e por ter acreditado em mim em todos meus momentos de dúvida.

Agradeço aos meus professores do ensino médio e do técnico, que despertaram meu amor pelo conhecimento e fortaleceram meu interesse pelo curso de computação.

Agradeço também aos meus professores da graduação pelo conhecimento e pelas oportunidades proporcionadas. Em especial, ao Prof. Dr. João Paulo Papa, meu orientador.

Agradeço aos meus amigos do curso pelos bons momentos que tivemos e por terem feito mais alegres nossas aulas e nossos trabalhos. Em especial, quero agradecer a Davi Augusto Neves Leite, Luis Henrique Morelli e Luiz Fernando Merli de Oliveira Sementille.

Por fim, agradeço a todos que contribuíram de alguma forma com minha formação.

"A ciência é o processo que nos leva da confusão ao entendimento."

Brian Greene

Resumo

Nas últimas décadas, a demanda por uma melhor administração de recursos como tempo e dinheiro tem crescido na sociedade, fazendo com que problemas de otimização sejam cada vez mais estudados por áreas como pesquisa operacional e ciência da computação. No entanto, as técnicas tradicionais de otimização não são eficientes para lidar com os problemas desafiadores encontrados no mundo real. Nesse cenário, as meta-heurísticas se apresentam como uma alternativa interessante, uma vez que são capazes de encontrar soluções satisfatórias para inúmeros problemas. Logo, faz-se necessário o desenvolvimento de uma ferramenta que traga as mais populares entre essas abundantes técnicas, acompanhadas de distintos problemas artificiais de teste que traduzam suas performances, viabilizando a rápida realização de análises comparativas. Neste trabalho, estudou-se as meta-heurísticas e as funções de teste mais empregadas na literatura, visando a criação de um ambiente web. Dentre as funcionalidades dessa aplicação, pode-se citar o ajuste de parâmetros de execução, a apresentação do progresso das tarefas em tempo real, a persistência dos dados de execução, a apresentação dos resultados juntamente com dados estatísticos e a produção de gráficos de convergência.

Palavras-chave: Aplicação Web; Análise Comparativa; Meta-heurísticas; Funções de Teste; Otimização Matemática.

Abstract

In recent years, the demand for better management of resources like time and money has grown, making optimization problems increasingly studied by areas such as operations research and computer science. However, traditional optimization methods are inefficient in dealing with challenging issues that can be found in the real world. In this scenario, metaheuristics are an exciting alternative, since they can solve numerous problems satisfactorily. Therefore, it is crucial to develop a tool that brings the most popular among these abundant methods and different artificial test problems that translate their performances, allowing quick comparative analyses. In this work, the meta-heuristics and the test functions most used in the literature are studied to create a web environment. Among the functionalities of this application, it is possible to mention the setting of execution parameters, the presentation of the progress of the tasks in real-time, the persistence of the execution data, the production of results with statistical data and the plotting of convergence.

Keywords: Web Application; Comparative Analysis; Meta-heuristics; Test Functions; Mathematical Optimization.

Lista de Figuras

Figura 1 – Login para Administradores.	36
Figura 2 – Página Inicial da Administração.	37
Figura 3 – Usuários Cadastrados.	38
Figura 4 – Otimizadores Cadastrados.	38
Figura 5 – Funções de Teste Cadastradas.	39
Figura 6 – Tarefas Cadastradas.	40
Figura 7 – Associação das Tarefas aos seus Proprietários.	41
Figura 8 – Login para Usuários Comuns.	42
Figura 9 – Login com Erro de Campos Vazios.	43
Figura 10 – Login com Erro de Dados Inválidos.	43
Figura 11 – Cadastro de Usuários Comuns.	44
Figura 12 – Cadastro com <i>Tooltip</i>	45
Figura 13 – Cadastro com Erro de Campos Vazios.	46
Figura 14 – Cadastro com Erro de Usuário Inválido.	47
Figura 15 – Cadastro com Erro de Senhas Distintas.	48
Figura 16 – Login com Mensagem de Usuário Criado.	49
Figura 17 – Redefinição de Senha.	50
Figura 18 – Redefinição com Erro de E-mail Não Encontrado.	50
Figura 19 – Redefinição com Mensagem de E-mail Enviado.	51
Figura 20 – E-mail de Redefinição Recebido pelo Usuário.	51
Figura 21 – Alteração de Senha.	52
Figura 22 – Login com Mensagem de Senha Alterada.	53
Figura 23 – <i>Dashboard</i>	54
Figura 24 – Tabela de Tarefas da <i>Dashboard</i>	54
Figura 25 – Nova Tarefa de Otimização.	56
Figura 26 – Nova Tarefa de Otimização com Otimizadores.	57
Figura 27 – Nova Tarefa de Otimização com Funções.	58
Figura 28 – Nova Tarefa de Otimização com Erro.	59
Figura 29 – Nova Tarefa de Otimização com Função Renderizada.	60
Figura 30 – Parte 1 dos Detalhes da Tarefa.	61
Figura 31 – Parte 2 dos Detalhes da Tarefa.	61
Figura 32 – Barra de Progresso da Tarefa Cancelada.	62
Figura 33 – Barra de Progresso da Tarefa Concluída.	62
Figura 34 – Principais Resultados da Tarefa.	62
Figura 35 – Resultados da Otimização.	63
Figura 36 – Gráfico de Convergência.	64

Figura 37 – Resultados da Otimização Com Estatísticas. 65

Figura 38 – Gráfico de Convergência Média. 65

Figura 39 – Página de Erro 404 66

Lista de Abreviaturas e Siglas

ABC	Artificial Bee Colony
CS	Cuckoo Search
CSS	Cascading Style Sheets
FA	Firefly Algorithm
GA	Genetic Algorithm
HTML	HyperText Markup Language
JS	JavaScript
MOF	Meta-heuristic Optimization Framework
MVC	Model-View-Controller
PSO	Particle Swarm Optimization
SA	Simulated Annealing

Sumário

1	INTRODUÇÃO	14
1.1	Problema	15
1.2	Justificativa	16
1.3	Objetivos	16
1.3.1	Objetivo Geral	17
1.3.2	Objetivos Específicos	17
1.4	Contribuição	17
1.5	Organização do Trabalho	18
2	FUNDAMENTAÇÃO TEÓRICA	19
2.1	Otimização Matemática	19
2.2	Meta-heurísticas	21
2.2.1	Algoritmo Genético	22
2.2.2	Algoritmo do Vaga-lume	22
2.2.3	Busca Cuco	22
2.2.4	Colônia Artificial de Abelhas	23
2.2.5	Otimização por Enxame de Partículas	23
2.2.6	Recozimento Simulado	23
2.3	Funções de Teste	24
2.3.1	Powell Sum	25
2.3.2	Rosenbrock	25
2.3.3	Step	25
2.3.4	Ackley 1	25
2.3.5	Alpine 1	25
2.3.6	Csendes	26
2.3.7	Griewank	26
2.3.8	Rastrigin	26
2.3.9	Salomon	26
2.3.10	Schwefel	26
2.4	Aplicações Web	27
2.4.1	Padrão MVC	28
3	METODOLOGIA	29
3.1	Escolha das Meta-heurísticas	29
3.2	Escolha das Funções de Teste	29
3.3	Etapas do Desenvolvimento	30

3.4	Ferramentas	31
3.4.1	Opytimizer	31
3.4.2	Opytimark	32
3.4.3	Django	32
3.4.4	Celery e Redis	32
3.4.5	MathJax	33
3.4.6	Plotly	33
3.4.7	Bootstrap	33
3.5	Métricas	34
4	DESENVOLVIMENTO	35
4.1	Painel Administrativo	35
4.1.1	Login	35
4.1.2	Página Inicial	36
4.1.2.1	Lista de Usuários	37
4.1.2.2	Lista de Otimizadores	38
4.1.2.3	Lista de Funções de Teste	39
4.1.2.4	Lista de Tarefas	39
4.1.2.5	Lista de Tarefas dos Usuários	40
4.2	MetaOPT	41
4.2.1	Login	41
4.2.2	Cadastro	44
4.2.3	Redefinição de Senha	49
4.2.4	<i>Dashboard</i>	53
4.2.5	Nova Tarefa de Otimização	54
4.2.6	Detalhes da Tarefa	60
4.2.7	Resultados	62
4.2.7.1	Execução Única	63
4.2.7.2	Múltiplas Execuções	64
4.2.8	<i>Página de Erro</i>	65
5	CONCLUSÃO	67
5.1	Trabalhos Futuros	67
	REFERÊNCIAS	69

1 Introdução

De modo geral, tarefas de otimização fazem parte do cotidiano dos seres humanos, seja em âmbito pessoal ou profissional. Segundo [Rao \(2009\)](#), pode-se entender por otimização o ato de obter a melhor solução para um determinado problema levando em conta as circunstâncias impostas. Para [Yang \(2014\)](#), trata-se de uma forma de planejamento, uma vez que recursos como tempo e dinheiro são valiosos e limitados, o que cria uma forte necessidade de aproveitá-los da melhor forma possível. Ademais, esse autor argumenta que a otimização está presente tanto na organização de uma atividade pessoal, como um itinerário para as férias da família, quanto em atividades de pesquisa operacional desempenhadas por engenheiros de produção.

[Rao \(2009\)](#) afirma que as empresas, por meio da otimização, visam minimizar os custos e maximizar os ganhos. Esses atos representam diferentes tarefas de otimização. De acordo com [Yang \(2014\)](#), os problemas de otimização são organizados matematicamente por meio de funções e equações. Em síntese, pode-se modelar um custo de produção por meio de uma função objetivo e minimizar essa função. No entanto, para minimizar, deve-se também considerar as restrições do problema. Tais restrições são descritas por equações com as variáveis de decisão que constituem a função objetivo. Sendo assim, otimizar o problema consiste em encontrar os melhores valores para as variáveis de decisão. Por melhores valores, entende-se a solução que respeite as restrições impostas e resulte no menor valor possível da função objetivo.

Para encontrar a solução ótima de um problema, [Yang \(2014\)](#) alega que se pode aplicar diversos métodos ou técnicas de otimização. Entretanto, como [Rao \(2009\)](#) observa, as técnicas tradicionais, que empregam cálculo ou métodos numéricos, não são ideais para os problemas complexos do mundo real que engenheiros frequentemente precisam resolver. Por esse motivo, os otimizadores meta-heurísticos surgem como uma alternativa bem interessante. Segundo [Yang \(2014\)](#), as meta-heurísticas são capazes de produzir soluções agradáveis por meio de tentativa e erro, sem exigir que as funções sejam contínuas ou diferenciáveis e fazendo uso somente do valor da função. Embora não garantam que a solução ótima seja encontrada, empregam um grau de randomização que permite mesclar buscas globais e locais. A partir desse mecanismo, pressupõe-se que, com tempo suficiente, uma solução satisfatória seja encontrada.

Outro motivo do bom desempenho dessas técnicas é que seus projetistas buscam inspiração em mecanismos de resolução de problemas já existentes. De acordo com [Yang \(2014\)](#), há uma tendência de abstrair comportamentos observados na natureza como a Teoria Evolucionista de Darwin ou a alimentação, locomoção e organização de certas espécies de seres vivos. A ideia é construir algoritmos análogos utilizando uma população de soluções. Nesse sentido, a vantagem é que muitas soluções são manipuladas ao mesmo tempo, como se a busca por uma solução fosse uma caça ao tesouro com vários caçadores. A busca é iniciada em

um local aleatório e, com o tempo, o caçador move-se para um local mais próximo de uma solução. Nesse processo, os caçadores podem compartilhar informações para que uma área maior de busca seja coberta e vários deles se aproximem de regiões interessantes. Em alguns casos, pode-se, com o tempo, manter os melhores, remover alguns e introduzir novos, o que introduz uma certa variabilidade na população. Conforme o autor, algoritmos que fazem uso de uma população de agentes que compartilha informações para encontrar a solução baseiam-se na Inteligência dos Exames, enquanto os algoritmos que introduzem a variabilidade para evoluir a população baseiam-se na teoria de Darwin, sendo conhecidos como Algoritmos Evolutivos.

Além da engenharia, a área da computação tem precisado lidar com problemas difíceis também. A título de exemplo, [Yang \(2014\)](#) menciona a necessidade de otimizar os hiperparâmetros de modelos de aprendizado de máquina, problema complexo que tem sido resolvido com meta-heurísticas. No entanto, o autor lembra que essas técnicas bio-inspiradas, bem convenientes, precisam ser testadas para se descobrir a mais apropriada para cada tipo de problema. É claro que essas técnicas não estão limitadas a tipos específicos de problemas, porém algumas podem lidar com mais rapidez ou facilidade com certos problemas. Para mais, [Yang \(2014\)](#) evidencia que a quantidade de métodos heurísticos que têm surgido nos últimos anos é exorbitante. Nesse contexto, [Rao \(2009\)](#) e [Yang \(2014\)](#) concordam que é preciso analisar qual heurística lida melhor com certos tipos de problemas levando em conta o tempo de convergência, o número de soluções desejadas entre outras particularidades.

1.1 Problema

Nesse cenário, destaca-se os módulos que trazem a implementação das meta-heurísticas mais populares para que sejam rapidamente usadas por profissionais que precisam tomar decisões de forma inteligente. Tanto [Parejo et al. \(2012\)](#) quanto [Ramírez, Barbudo e Romero \(2021\)](#) elencam as mais populares dessas, denominando-as Frameworks de Meta-heurísticas de Otimização, ou Metaheuristic Optimization Frameworks (MOFs) no original. Nesses estudos, os autores realizaram uma avaliação dessas ferramentas de acordo com suas funcionalidades e experiências de uso. Dessa análise, concluiu-se que essas ferramentas precisavam melhorar no quesito de interface gráfica, uma vez que poucas permitiam a seleção dos otimizadores e das funções objetivo, a configuração dos parâmetros de execução e a produção de gráficos de convergência sem o uso de código ou programas complementares.

Outro ponto interessante extraído da análise é que essas ferramentas precisam ser instaladas na máquina do usuário. Em alguns casos, esses instrumentos dependem de um certo sistema operacional ou de certos programas. Dentre as ferramentas avaliadas por [Parejo et al. \(2012\)](#) e as avaliadas por [Ramírez, Barbudo e Romero \(2021\)](#), pode-se notar a ausência de interfaces web de fácil uso e que não demandem conhecimento técnico ou instalação de componentes. Com uma interface assim, qualquer principiante poderia rapidamente testar e

comparar as técnicas com funções de teste comumente empregadas em análises de performance.

1.2 Justificativa

Por conta das inconveniências expostas, surgiu o projeto de uma aplicação web para execução de meta-heurísticas de otimização por parte do Recogna¹, laboratório que atua nas áreas de inteligência artificial, otimização e biometria, e faz parte da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" em Bauru. Com base em sua concepção, esse projeto, denominado MetaOPT, constitui-se de dois módulos. O primeiro é o módulo de otimização, no qual se emprega meta-heurísticas populares na resolução de problemas artificiais para fins de comparação de performance. Já o segundo é o módulo de seleção de características, no qual se utiliza as técnicas para extrair as melhores características de bases de dados selecionadas também para fins de comparação de desempenho.

Nesse âmbito, o presente trabalho assume como projeto a implementação do módulo de otimização do MetaOPT. Além do mais, preocupa-se em relatar as atividades que sobressaem o escopo desse módulo, mas que estão relacionadas direta ou indiretamente com as tarefas de otimização e, por esse motivo, contaram com participação ativa do autor do trabalho não só no planejamento mas também no desenvolvimento e em consequentes deliberações.

Tanto no quesito da disponibilidade pela internet quanto no aspecto das comodidades de uma interface gráfica para os usuários, pode-se afirmar que a existência de uma ferramenta mais acessível para a execução de meta-heurísticas de otimização é indispensável. Com essa ferramenta, estudantes ou profissionais que precisam encontrar soluções boas para problemas operacionais, como engenheiros e cientistas da computação, podem empregar as meta-heurísticas para atacar certos problemas, avaliar o desempenho e, ainda, comparar os resultados, sem precisar de conhecimento prévio sobre a implementação das técnicas ou ter que instalar algum programa em suas máquinas. Ademais, para os novatos, a identificação do método mais conveniente para suas necessidades torna-se muito mais viável.

1.3 Objetivos

O presente trabalho tenciona criar uma interface para a internet voltada para estudantes ou profissionais que queiram ganhar familiaridade com as meta-heurísticas de otimização. Nesse sentido, comenta-se com mais rigor a meta estabelecida para o projeto na [subseção 1.3.1](#), enquanto, na [subseção 1.3.2](#), elenca-se os passos necessários para que essa meta seja atingida.

¹ Disponível em: <<https://www.recogna.tech/>>. Acesso em: 23 nov. 2022

1.3.1 Objetivo Geral

Desenvolver uma aplicação web para a fácil execução de tarefas de otimização com meta-heurísticas, viabilizando a realização de análises comparativas desses otimizadores a partir de qualquer dispositivo com conexão à internet, sem necessidade de instalar dependências ou possuir conhecimento técnico, disponibilizando funções de teste comumente empregadas no *benchmarking* dessas técnicas, permitindo o ajuste de parâmetros de execução, e apresentando, ao fim das execuções, os dados estatísticos e o gráfico de convergência.

1.3.2 Objetivos Específicos

Com base no propósito estabelecido, lista-se os seguintes objetivos específicos:

- Estudar as mais populares meta-heurísticas de otimização;
- Elencar as funções de teste comumente empregadas na análise de desempenho das heurísticas de otimização;
- Identificar as bibliotecas e demais tecnologias a serem empregadas no desenvolvimento da aplicação proposta;
- Determinar as entradas que a interface deve conter, como parâmetros de execução;
- Estabelecer as saídas devolvidas pela aplicação, como os dados estatísticos das execuções e o gráfico de convergência;
- Criar a interface de acordo com as tecnologias escolhidas;
- Integrar as bibliotecas responsáveis pela parte lógica da aplicação com a interface;
- Executar testes para validar o produto final em um servidor local.

1.4 Contribuição

Tendo em vista a abundância de meta-heurísticas de otimização na literatura, uma das contribuições deste trabalho é a identificação das técnicas mais populares. Ademais, de maneira análoga, o trabalho contribui para o levantamento das funções de teste de performance mais empregadas na literatura. Por fim, sua maior contribuição é a construção de um ambiente web para execução de tarefas de otimização, que não só auxiliará mas também facilitará o estudo de pessoas que queiram trabalhar com os otimizadores meta-heurísticos.

1.5 Organização do Trabalho

O presente trabalho está estruturado da seguinte maneira:

Capítulo 2: Apresenta conceitos como Otimização Matemática, Meta-heurísticas e Funções de Teste, além de explicar as técnicas e as funções empregadas no projeto;

Capítulo 3: Enumera as etapas que orientaram o desenvolvimento do trabalho, descreve as ferramentas utilizadas e cita as métricas apresentadas;

Capítulo 4: Detalha as etapas do desenvolvimento e apresenta a aplicação;

Capítulo 5: Por fim, conclui a monografia e evidencia possíveis trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, apresenta-se os conceitos relevantes para o projeto, tais como Otimização Matemática, Meta-heurísticas, Funções de Teste e Aplicações Web. Ademais, evidencia-se os otimizadores meta-heurísticos, os problemas artificiais para teste de performance e o padrão de arquitetura de software selecionados para a estruturação do aplicativo web proposto.

2.1 Otimização Matemática

A otimização é uma área de estudo que visa encontrar soluções para um certo problema modelado na forma de funções matemáticas com ou sem restrições, tendo como objetivo minimizar um custo ou maximizar um lucro. Neste trabalho, convencionou-se a minimização. Logo, um problema de otimização qualquer pode ser expresso conforme a [Equação 2.1](#).

$$\begin{aligned} &\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimizar}} && f_i(\mathbf{x}), (i = 1, 2, \dots, M), \\ &\text{sujeito a} && h_j(\mathbf{x}) = 0, (j = 1, 2, \dots, J), \\ &&& g_k(\mathbf{x}) \leq 0, (k = 1, 2, \dots, K), \end{aligned} \quad (2.1)$$

onde $f_i(\mathbf{x})$, $h_j(\mathbf{x})$ e $g_k(\mathbf{x})$ são funções de

$$\mathbf{x} = (x_1, x_2, \dots, x_n)^T. \quad (2.2)$$

O vetor \mathbf{x} apresentado na [Equação 2.2](#) abriga as variáveis de decisão do problema, que podem ser duas ou mais, constituindo um problema multivariável ($n > 1$), ou uma única variável, formando um problema monovariável ($n = 1$). Já as funções f_i da equação [Equação 2.1](#) são as funções objetivo. Caso sejam mais de uma, tem-se problemas multiobjetivo, caso seja uma única função, tem-se problemas mono-objetivo. Além disso, é importante mencionar que essas funções também são denominadas funções de custo. Por fim, as funções h_j e g_k representam, respectivamente, as restrições de igualdade e de desigualdade do problema, sendo condições que devem ser obedecidas pelas soluções candidatas das funções de custo.

Em suma, os valores que o vetor \mathbf{x} pode assumir durante a tarefa de otimização formam um conjunto denominado de espaço de busca. Por outro lado, o domínio de uma função objetivo f determina o espaço das soluções. Quando esse domínio é conhecido, o espaço de busca e o espaço de soluções podem coincidir, já que não há interesse em vetores fora do espaço de soluções. Por fim, há também o espaço das soluções viáveis \mathbb{V} , denominado região de viabilidade, que é a zona do espaço das soluções que respeita as restrições impostas. Caso o problema não possua restrições ($J = 0, K = 0$), ele é classificado como irrestrito.

Tendo em vista o exposto, pode-se dizer que o objetivo da otimização é encontrar a melhor solução possível para um problema. Por melhor, entende-se um $\mathbf{x}^* \in \mathbb{V}$ para o qual seja verdade a regra $\forall \mathbf{x} \in \mathbb{V}, f(\mathbf{x}^*) \leq f(\mathbf{x})$, isto é, \mathbf{x}^* resulta no menor valor de f dentre todos os vetores do espaço de soluções viáveis \mathbb{V} . Como se pode notar na condição, é possível que exista outro vetor que leve ao mesmo valor de f e, neste caso, há duas soluções ótimas para o problema. Também é possível que exista um subconjunto de \mathbb{V} no qual exista um $\mathbf{x}' \neq \mathbf{x}^*$ tal que $\forall \mathbf{x} \in \mathbb{V}, f(\mathbf{x}') \leq f(\mathbf{x})$. Nesse caso, \mathbf{x}' é dito ótimo local, vetor que confunde otimizadores, que, em geral, estão interessados no vetor de ótimo global \mathbf{x}^* .

Nesse contexto, [Rao \(2009\)](#) lembra o conceito da unimodalidade. De acordo com o autor, quando uma função possui um único pico, que representa um valor de máximo, ou um único vale, que representa um valor de mínimo, a função é dita unimodal. Caso a função possua mais de um pico ou vale, ela é dita multimodal. Problemas unimodais são mais simples de serem resolvidos, uma vez que a função possui um único ótimo, que é local e global ao mesmo tempo. O autor explica que os otimizadores podem se basear nessa suposição e procurar pelo vale onde está o valor ótimo. No entanto, se a função é multimodal e possui muitos vales, esses métodos deixam de ser apropriados, pois podem ficar presos nos ótimos locais.

Para encontrar o \mathbf{x}^* , ótimo global do problema, pode-se aplicar diversos métodos de otimização. De acordo com [Yang \(2014\)](#), há duas abordagens, as tradicionais e as não tradicionais. O autor afirma que, de modo geral, essas abordagens partem de um palpite inicial e atualizam esse valor dentro da região de viabilidade até que um certo critério de parada seja atingido. Esse critério pode ser um número de iterações ou um certo valor de *fitness*, que pode ser o custo a ser minimizado ou um indicativo desse custo. Contudo, a solução ótima pode ou não ser obtida, dependendo, por exemplo, de condições do método escolhido.

As técnicas tradicionais são constituídas por algoritmos determinísticos, enquanto as técnicas mais modernas são formadas por algoritmos estocásticos. Segundo [Yang \(2014\)](#), os algoritmos determinísticos podem utilizar o gradiente da função, o valor da função ou os dois para encontrar uma solução. Além disso, tais algoritmos possuem condições que devem ser respeitadas para que possam ser aplicados. Dessa forma, essas técnicas podem não performar bem para certas classes de funções, enquanto desempenham bem para outras classes.

Conforme complementa [Rao \(2009\)](#), os métodos tradicionais possuem um alcance limitado, uma vez que são úteis para problemas contínuos e diferenciáveis, enquanto os problemas do mundo real são complexos e nem sempre atendem a esses critérios. Ademais, como o autor aponta, esses métodos também fazem suposições quanto ao tipo do problema, havendo métodos específicos para problemas monovariáveis, multivariáveis, lineares, não lineares etc. Por fim, para que uma solução ótima ou quase ótima seja encontrada, é preciso que algumas condições sejam atendidas pelo problema, o que nem sempre é possível.

No entanto, pode-se afirmar que essas técnicas seguem passos concretos visando garantir a obtenção de uma solução ótima após um certo critério de parada, se suas condições

forem respeitadas. Por outro lado, as técnicas estocásticas nada mais são do que heurísticas ou meta-heurísticas de otimização. Tais técnicas são explicadas a seguir na [seção 2.2](#).

2.2 Meta-heurísticas

Esse grupo é menos exigente que o anterior, porém não oferece garantias. [Yang \(2014\)](#) afirma que heurísticas ou meta-heurísticas, termos intercambiáveis, designam algoritmos que produzem soluções agradáveis por meio de tentativa e erro, sem garantia de que uma solução ótima seja encontrada. Em síntese, essas técnicas utilizam somente o valor da função e empregam algum grau de randomização para executar buscas globais, que compõem a diversificação, e buscas locais, que constituem a intensificação. A combinação desses estágios evita que o algoritmo fique preso em um ótimo local, porém não garante que o global seja encontrado, podendo-se esperar, no mínimo, uma solução satisfatória.

É importante notar que uma tendência muito forte dessa subárea da otimização é formular algoritmos que abstraem comportamentos encontrados na natureza em seus mecanismos de busca de soluções. De acordo com [Yang \(2014\)](#), as meta-heurísticas populares e amplamente empregadas vieram dessa inclinação. Um dos subgrupos de heurísticas de otimização, por exemplo, são os algoritmos evolucionários, que empregam conceitos da teoria de evolução darwinista para aperfeiçoar uma população de soluções no decorrer de um número de gerações.

Segundo [Rao \(2009\)](#), métodos bio-inspirados tem sido mais empregados para lidar com tarefas do mundo real, como em problemas da engenharia. O autor afirma que isto se deve ao fato de as meta-heurísticas demandarem somente os valores da função ao invés do gradiente, que é a derivada da função para um certo x . Ademais, tais algoritmos não fazem suposições sobre a função, podendo ser aplicados para diversos tipos. Entretanto, [Yang \(2014\)](#) sugere que há estudos sobre a eficiência desses métodos em diferentes tipos de problemas.

Além dos valores da função, essas técnicas podem fazer uso de hiperparâmetros constantes e randômicos. Como apontado tanto por [Rao \(2009\)](#) quanto [Yang \(2014\)](#), os hiperparâmetros randômicos são obtidos de uma distribuição uniforme de números pseudoaleatórios, enquanto as constantes precisam ser atribuídas, cabendo ao usuário experimentar quais valores fazem o otimizador performar melhor. No entanto, os autores indicam que observações empíricas, realizadas pela academia, levaram a valores recomendados para essas constantes. É importante evidenciar que este trabalho optou pelos valores pré-calculados para as constantes, pois já estão presentes na biblioteca de meta-heurísticas mencionada no [Capítulo 3](#).

Nas subseções a seguir, comenta-se os otimizadores relevantes para o projeto de acordo com seus respectivos idealizadores, visando explicitar os mecanismos de busca propostos.

2.2.1 Algoritmo Genético

[Holland \(1992\)](#) explica que propôs o Algoritmo Genético, Genetic Algorithm (GA) no inglês, por volta da década de 70 durante suas pesquisas sobre sistemas artificiais. De acordo com o autor, a abordagem foi concebida pela abstração de conceitos da teoria darwinista de evolução por seleção natural. Em síntese, as soluções são codificadas em vetores de dígitos binários nos quais se aplicam operações evolutivas como cruzamento, mutação e seleção.

Em suma, a população desse Algoritmo Evolutivo é inicializada randomicamente e evolvida por um certo número de gerações. A cada geração, podem ocorrer alguns cruzamentos, que consistem na troca de partes de dois vetores. Ademais, podem ocorrer mutações, que consistem em inversões de bits dos vetores de solução. Por último, ao final das gerações, o operador de seleção é aplicando, formando uma nova população através das melhores soluções.

2.2.2 Algoritmo do Vaga-lume

A técnica conhecida por Algoritmo do Vaga-lume, ou Firefly Algorithm (FA) no original, foi criada por [Yang \(2009\)](#). Conforme o autor explica, essa heurística baseada em população abstrai o comportamento dos vaga-lumes com suas piscadas. Uma das prováveis justificativas para as piscadas é atrair parceiros para o acasalamento, logo o algoritmo assume uma população unissex de vaga-lumes que se atraem um pelo outro de acordo com a intensidade da luz das piscadas, sendo que o vaga-lume com a luz mais intensa é o mais próximo de uma solução.

Dessa forma, os vaga-lumes com intensidade mais baixa se movem direção aos vaga-lumes com maior intensidade de luz. No entanto, para permitir que a diversificação ocorra, a atratividade varia de acordo com a distância entre os vaga-lumes. Sendo assim, um vaga-lume é menos atraído e, portanto, se move menos em direção a vaga-lumes mais distantes.

2.2.3 Busca Cuco

O algoritmo da Busca Cuco, ou Cuckoo Search (CS) no inglês, foi criado em 2009 por [Yang e Deb \(2009\)](#), motivados pelo parasitismo de ninhada empregado por certas espécies de cuco. Esses pássaros depositam seus ovos nos ninhos de outras espécies, que podem sequer diferenciar os novos ovos e acabam muitas vezes criando os cucos como filhos. Em alguns casos, quando reconhecem o intruso, as espécies se livram dele ou abandonam o ninho.

Essa é uma técnica baseada em população, na qual cada agente representa a posição de um ninho com o ovo de um cuco. A cada iteração, o cuco voa aleatoriamente e deposita um ovo. Em seguida, um dos ninhos com ovos é selecionado, se a nova posição do ovo for melhor, o novo ovo é colocado no ninho. Ao fim do processo, parte dos ninhos ruins são eliminados, enquanto outros são criados para compor a nova população, junto com os melhores ninhos. Ademais, é possível que a espécie parasitada descubra o ovo do cuco e se livre do ninho.

2.2.4 Colônia Artificial de Abelhas

Segundo [Karaboga e Basturk \(2007\)](#), a Colônia Artificial de Abelhas, ou Artificial Bee Colony (ABC) no original, é uma técnica baseada em população que abstrai a inteligência do forrageamento dos enxames de abelhas produtoras de mel. Dessa forma, pertence ao grupo de algoritmos que fazem uso da Inteligência de Enxame, conceito descrito na [subseção 2.2.5](#).

No algoritmo, há três tipos de abelha. Cada abelha trabalhadora toma conta de uma fonte de alimento e identifica fontes vizinhas. Por sua vez, uma abelha exploradora procura aleatoriamente por novas fontes de alimento. Por fim, as abelhas oportunistas recebem informações das abelhas trabalhadoras para que possam escolher uma fonte de alimento. Além disso, após um período de tempo em uma fonte de alimento, as abelhas podem deixá-la para procurar fontes melhores. Sendo assim, a função objetivo precisa ser codificada para indicar a quantidade de néctar em uma fonte de alimento. Isso deve ser feito para que a fonte com mais néctar indique o vetor que resulte no menor valor da função.

2.2.5 Otimização por Enxame de Partículas

A Otimização por Enxame de Partículas, ou Particle Swarm Optimization (PSO) no inglês, é fruto do trabalho de [Kennedy e Eberhart \(1995\)](#). De acordo com os autores, essa técnica baseada em população foi inspirada no comportamento de colônias de seres vivos, como peixes e pássaros que se agrupam em cardumes e bandos respectivamente. Ademais, cunhou o conceito da Inteligência de Enxame, que consiste na troca de informações entre os agentes com a finalidade de direcionar o enxame de uma maneira mais adequada.

De forma resumida, o mecanismo de busca do PSO procura pela solução da função custo movendo as partículas pelo espaço com base em informações como a melhor posição do enxame, a melhor posição da partícula e a posição atual da partícula. A ideia é que as partículas sejam atraídas pelas posições mencionadas, além de se mover um pouco de forma aleatória. Com o tempo, o enxame tende a se concentrar ao redor de uma solução satisfatória.

2.2.6 Recozimento Simulado

O Recozimento Simulado, ou Simulated Annealing (SA) no original, é uma das meta-heurísticas mais antigas e foi proposta por [Kirkpatrick, Gelatt e Vecchi \(1983\)](#). Trata-se de um algoritmo de busca aleatória baseado em trajetória. Em síntese, simula o processo de recozimento no qual um metal é superaquecido e, em seguida, resfriado lentamente.

Quanto maior a temperatura do metal, com mais velocidade os átomos se locomovem. Ao resfriar o metal lentamente, os átomos passam a se locomover mais devagar. No processo de recozimento, os átomos se organizam em uma estrutura cristalina de mínima energia. O algoritmo tenta capturar essa ideia, logo possui um parâmetro de temperatura que controla a

movimentação das partículas, que é acentuada no início e diminui conforme a temperatura decai. Com aleatoriedade o suficiente, espera-se que uma solução ótima seja encontrada.

O funcionamento pode ser simplificado por meio da metáfora das bolas saltitantes. Se bolas saltitantes fossem soltas em uma superfície, pulassem e já perdessem energia, ficariam presas em algum vale, que poderia ser um ponto de mínimo local. Porém, se as bolas perdessem energia lentamente, algumas poderiam eventualmente cair em um ponto de mínimo global.

2.3 Funções de Teste

Uma função de teste, ou de *benchmarking*, é um problema artificial que pode ser utilizado para avaliar a performance de algoritmos de otimização. De acordo com [Hussain et al. \(2017\)](#), essas funções podem ser agrupadas de acordo com características como a modalidade, a separabilidade e a dimensão. Em suma, as funções de teste podem ser unimodais ou multimodais, separáveis ou não separáveis, e ter uma ou mais variáveis de decisão.

As funções unimodais possuem um único ponto de ótimo, que é o ótimo global. Já as funções multimodais possuem ótimos locais e um verdadeiro ponto de ótimo, que é o global. [Hussain et al. \(2017\)](#) explica que as funções unimodais são úteis para testar a busca local dos otimizadores, enquanto as multimodais procuram testar a busca global, já que é necessário explorar mais os mínimos locais do espaço até encontrar o ótimo verdadeiro.

Em relação à separabilidade, os autores afirmam que em algumas dessas funções há variáveis que não têm relação com as demais, sendo possível procurar os valores para essas variáveis com mais liberdade, como se fossem várias funções sendo otimizadas em paralelo. Essas funções separáveis são mais fáceis de resolver. Por outro lado, os autores afirmam que as funções nas quais as variáveis possuem fortes relações com as demais são mais difíceis.

Por fim, em relação à dimensão, os autores afirmam que muitas dessas funções possuem uma forma geral e podem ser definidas para diferentes valores de n . Conforme os autores, quanto maior o número de variáveis, maior o espaço de busca. Consequentemente, o otimizador tem que lidar com mais ótimos locais. Destarte, funções com dimensões menores são fáceis de resolver, porém para avaliar a capacidade dos otimizadores é interessante usar várias dimensões.

Levando em conta o estabelecido, pode-se apresentar as funções pertinentes ao projeto. Para tal, faz-se necessário pontuar que o presente trabalho optou por problemas irrestritos de otimização global, logo o objetivo é encontrar o mínimo global dentro do espaço de soluções da função de custo. Sendo assim, comenta-se, com fundamento nas pesquisas de [Hussain et al. \(2017\)](#) e [Jamil e Yang \(2013\)](#), as funções de teste pertinentes ao projeto.

2.3.1 Powell Sum

A Powell Sum é uma função contínua, diferenciável, separável e unimodal. Geralmente, é avaliada em $-1 \leq x_i \leq 1$ ($i = 1, 2, \dots, n$). Ademais, possui o ótimo global $\mathbf{x}^* = (0, 0, \dots, 0)$, que resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.3](#) apresenta a forma geral da função.

$$f(\mathbf{x}) = \sum_{i=1}^n |x_i|^{i+1} \quad (2.3)$$

2.3.2 Rosenbrock

A Rosenbrock é uma função contínua, diferenciável, não separável e unimodal. Costuma ser avaliada em $-30 \leq x_i \leq 30$ ($i = 1, 2, \dots, n$). Para mais, seu ótimo global é $\mathbf{x}^* = (1, 1, \dots, 1)$ e resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.4](#) mostra sua fórmula geral.

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \quad (2.4)$$

2.3.3 Step

A Função Step, ou Degrau, é descontínua, não diferenciável, separável e unimodal. Além disso, é avaliada em $-100 \leq x_i \leq 100$ ($i = 1, 2, \dots, n$), tendo o ótimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ e valor ótimo $f(\mathbf{x}^*) = 0$. A [Equação 2.5](#) exibe sua expressão geral.

$$f(\mathbf{x}) = \sum_{i=1}^n |x_i| \quad (2.5)$$

2.3.4 Ackley 1

A Ackley 1 é uma função contínua, diferenciável, não separável e multimodal. Geralmente, é avaliada em $-32 \leq x_i \leq 32$ ($i = 1, 2, \dots, n$). Possui um ótimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ que resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.6](#) mostra sua forma geral.

$$f(\mathbf{x}) = -20e^{-0,2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi x_i)} + 20 + e \quad (2.6)$$

2.3.5 Alpine 1

A Alpine 1 é uma função contínua, não diferenciável, separável e multimodal. Costuma ser avaliada em $-10 \leq x_i \leq 10$ ($i = 1, 2, \dots, n$). Ademais, possui um ótimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ que resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.7](#) exibe sua fórmula geral.

$$f(\mathbf{x}) = \sum_{i=1}^n |x_i \sin(x_i) + 0,1x_i| \quad (2.7)$$

2.3.6 Csendes

A Csendes é uma função contínua, diferenciável, separável e multimodal. Geralmente, é avaliada em $-1 \leq x_i \leq 1$ ($i = 1, 2, \dots, n$). Para mais, possui um mínimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ que resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.8](#) mostra sua expressão geral.

$$f(\mathbf{x}) = \sum_{i=1}^n x_i^6 (2 + \sin(\frac{1}{x_i})) \quad (2.8)$$

2.3.7 Griewank

A Griewank é uma função contínua, diferenciável, não separável e multimodal. Ademais, é geralmente avaliada em $-100 \leq x_i \leq 100$ ($i = 1, 2, \dots, n$), e possui um mínimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ e valor ótimo $f(\mathbf{x}^*) = 0$. A [Equação 2.9](#) apresenta sua forma geral.

$$f(\mathbf{x}) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}}) \quad (2.9)$$

2.3.8 Rastrigin

A Rastrigin é uma função contínua, diferenciável, separável e multimodal. Costuma ser avaliada em $-5,12 \leq x_i \leq 5,12$ ($i = 1, 2, \dots, n$). Ademais, possui um ótimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ que resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.10](#) apresenta sua fórmula geral.

$$f(\mathbf{x}) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i)) \quad (2.10)$$

2.3.9 Salomon

A Salomon é uma função contínua, diferenciável, não separável e multimodal. Geralmente, é avaliada em $-100 \leq x_i \leq 100$ ($i = 1, 2, \dots, n$). Ademais, possui um mínimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ que resulta em $f(\mathbf{x}^*) = 0$. A [Equação 2.11](#) exibe sua expressão geral.

$$f(\mathbf{x}) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^n x_i^2}) + 0,1 \sqrt{\sum_{i=1}^n x_i^2} \quad (2.11)$$

2.3.10 Schwefel

A Schwefel é uma função contínua, diferenciável, parcialmente separável e multimodal. Para mais, é avaliada em $-10 \leq x_i \leq 10$ ($i = 1, 2, \dots, n$) e possui um mínimo global $\mathbf{x}^* = (0, 0, \dots, 0)$ que resulta em $f(\mathbf{x}^*) = 420,9687$. A [Equação 2.12](#) mostra sua fórmula.

$$f(\mathbf{x}) = 418,9829n - \sum_{i=1}^n x_i \sin(\sqrt{|x_i|}) \quad (2.12)$$

2.4 Aplicações Web

Segundo [Casteleyn et al. \(2009\)](#), a internet surgiu com o propósito de compartilhar informações por meio de documentos estáticos, porém esse ambiente foi enriquecido com o tempo. Os autores afirmam que novas tecnologias tornaram possível utilizar a web não só para apresentar conteúdo mas também para executar aplicações. Por esse motivo, nos dias atuais, serviços de comércio eletrônico, redes sociais, salas de aula virtuais, serviços corporativos e outras inimagináveis aplicações podem ser acessadas pela internet.

No entanto, como os autores ressaltam, criar uma aplicação web não é uma tarefa fácil, já que é preciso conhecer bem a estrutura e o funcionamento desse ambiente. Para o usuário, parece simples, pois só tem acesso à interface da aplicação. Ao contrário de um aplicativo local, é preciso trabalhar com uma arquitetura cliente-servidor e lidar com as camadas presentes nesse modelo. Entretanto, após essa habituação, as aplicações para web podem ser mais atraentes.

As aplicações na web, de acordo com os autores, proporcionam uma maior acessibilidade, já que os usuários podem utilizar os serviços do aplicativo por meio de qualquer dispositivo que possua conexão com a internet e tenha um navegador. Além do mais, elimina a necessidade de instalação do próprio aplicativo e de suas dependências. Outrossim, em decorrência da natureza mais leve do navegador, tarefas que seriam executadas localmente passam a processadas em um servidor remoto, exigindo poucos recursos da máquina do usuário.

O funcionamento desses sistemas é segmentado de acordo com as camadas de rede implementadas pelos sistemas operacionais dos dispositivos. [Casteleyn et al. \(2009\)](#) mencionam que as três camadas que comumente fazem parte dessa divisão são as camadas de dados, de aplicação e de apresentação. A camada de dados preocupa-se com os formatos de dados ou bancos de dados empregados. Na camada de aplicação, preocupa-se com a parte lógica bem como os caminhos que são disponibilizados para a navegação do usuário. Por fim, a camada de apresentação preocupa-se com a parte visual na qual as informações são exibidas ao usuário.

Tais camadas encontram-se distribuídas entre cliente e servidor. De modo resumido, o cliente faz uma requisição ao servidor, o servidor processa a requisição e devolve uma resposta ao usuário. Tendo em vista esse fluxo, pode-se dizer que tanto do lado do cliente quanto do servidor, as três camadas estão presentes. Do lado servidor, encontra-se a parte visual, as páginas da aplicação, as tarefas que são executadas pela aplicação e a base de dados. Já o cliente, ao fazer requisições, recebe páginas e dados do servidor, que podem ficar armazenados no navegador. Além do mais, algumas tarefas podem ser executadas no cliente.

Para acomodar tais camadas, pode-se modularizar a aplicação seguindo um padrão de arquitetura de software. O presente projeto optou pelo padrão Model-View-Controller (MVC). Logo, na [subseção 2.4.1](#), expõe-se, segundo [Casteleyn et al. \(2009\)](#), os itens desse modelo.

2.4.1 Padrão MVC

Esse padrão de arquitetura emprega três módulos para organizar o código da aplicação. Cada letra do nome do padrão indica um dos componentes, sendo M de Modelo, V de Visão e C de Controle. Essa divisão tem como base a separação de preocupações, que como o nome sugere, procura separar o código em módulos que só tomam conta de parte da aplicação.

Nessa estrutura, o Modelo fica responsável pela lógica de negócios da aplicação. Logo, trata-se da seção que realiza operações sobre a base de dados, além de executar demais tarefas relacionadas às funcionalidades do aplicativo. Entretanto, esse módulo não se preocupa com as requisições do usuário ou com a forma na qual os dados são apresentados ao usuário.

Por sua vez, o módulo Visão é o responsável por abrigar a lógica da interface de usuário do aplicativo, contendo os arquivos HTML que estruturam as páginas web. É importante mencionar que o módulo não se preocupa com o tipo da requisição realizada pelo usuário nem com a fonte dos dados. Sendo assim, única preocupação da Visão é formatar as páginas da aplicação e os dados contidos nessas páginas para apresentar ao usuário.

Por fim, o Controle se encarrega das interações do usuário com a aplicação, que ocorrem por meio da parte visual. Dessa forma, também possui a responsabilidade de invocar as operações do Modelo, recebendo os dados da base e repassando ao módulo Visão. Embora observe as escolhas tomadas pelo usuário por meio da Visão, esse módulo não se preocupa com a lógica de negócios contida nas operações do Modelo nem com as formas com as quais a Visão formata os dados para apresentá-los ao usuário por meio de suas páginas web.

3 Metodologia

Neste capítulo, descreve-se os critérios para seleção tanto das meta-heurísticas quanto das funções para *benchmarking*, as etapas que constituíram o desenvolvimento da aplicação proposta, as ferramentas que compõem a parte visual e a parte lógica e, por fim, as métricas que são apresentadas ao usuário após a execução das tarefas de otimização.

3.1 Escolha das Meta-heurísticas

Para selecionar as meta-heurísticas que compõem a aplicação, utilizou-se o estudo de [Dokeroglu et al. \(2019\)](#), no qual se mede a popularidade dessas técnicas de otimização. Nesse estudo, os autores dividem as técnicas em clássicas e nova geração. Na visão dos autores, considera-se da nova geração, as técnicas concebidas entre 2000 e 2020. Por conseguinte, os otimizadores propostos antes do ano 2000 enquadram-se no grupo de técnicas clássicas.

Tendo como base o exposto, em maio de 2019, os autores realizaram uma pesquisa por estudos envolvendo as meta-heurísticas clássicas e as da nova geração na base de dados do Google Scholar. Por meio dessa pesquisa, quantificou-se o número de trabalhos encontrados para 14 técnicas de cada grupo. Logo, pode-se dizer que obtiveram uma noção das 28 meta-heurísticas mais populares na literatura, tomando como referência a base do Scholar.

No caso das meta-heurísticas clássicas, os autores concluíram que a mais popular é a GA, com 1.270.000 estudos relacionados. Por sua vez, encontraram que a meta-heurística mais popular da nova geração é a ABC, com 37.400 artigos publicados. Através desses resultados, optou-se por selecionar três técnicas populares de cada grupo para compor os otimizadores do aplicativo, analisando as mais publicadas em direção às técnicas com menos publicações. Das clássicas, escolheu-se GA, PSO e SA, enquanto das mais recentes, selecionou-se FA, CS e ABC. É importante notar que a ideia geral desses algoritmos foi explanada no [Capítulo 2](#).

3.2 Escolha das Funções de Teste

De acordo com [Hussain et al. \(2017\)](#), o conjunto de teste para um otimizador meta-heurístico não possui uma convenção em relação ao número de funções ou ao tipo das funções. No entanto, os autores analisaram vários artigos, nos quais métodos foram propostos e avaliados, intentando obter uma noção das escolhas tomadas pelos projetistas. Dessa análise, os autores obtiveram diferentes números de funções, mas concluíram que o mais frequente foi dez. Sendo assim, o presente trabalho optou por formar um conjunto com dez problemas de teste.

Esses mesmos autores também observaram as configurações mais comuns em relação

à modalidade das funções. Dessa observação, extraíram que o mais comum é utilizar uma proporção maior de funções multimodais, concluindo que das dez funções, tamanho comum, o mais frequente é que por volta de seis sejam multimodais. Partindo daí, o presente projeto optou por sete funções multimodais e três unimodais para compor o conjunto de teste.

Tendo em mente o tamanho e proporção de cada tipo de função, analisou-se não só o estudo de [Hussain et al. \(2017\)](#) mas também o levantamento de [Jamil e Yang \(2013\)](#). No estudo de [Hussain et al. \(2017\)](#), são elencadas 20 das funções mais comuns para averiguar a performance das meta-heurísticas. Ademais, no levantamento de [Jamil e Yang \(2013\)](#), reúne-se as 175 funções de teste mais empregadas na literatura. Com base nessa análise, selecionou-se as funções unimodais Powell Sum, Rosenbrock e Step. Por sua vez, escolheu-se as funções multimodais Ackley 1, Alpine 1, Csendes, Griewank, Rastrigin, Salomon e Schwefel. Vale lembrar que essas funções já foram apresentadas em detalhes no [Capítulo 2](#).

3.3 Etapas do Desenvolvimento

O desenvolvimento da aplicação proposta pode ser organizado por etapas. Nesse sentido, iniciou-se definindo os campos que são apresentados ao usuário na página de execução de novas tarefas. Ademais, determinou-se as saídas que são apresentadas ao usuário após a execução, empregando as métricas comentadas na [seção 3.5](#). Também, decidiu-se quais informações da tarefa seriam persistidas no banco de dados para que o usuário pudesse acompanhar o progresso da execução ou conferir os resultados posteriormente. Ainda em relação às tarefas, deliberou-se sobre um tempo limite de execução para as tarefas e sobre o escalonamento das tarefas dos múltiplos usuários da aplicação. Outrossim, tendo como base a gerência das tarefas, fez-se necessário pensar em cadastros de usuário, aspirando o estabelecimento de uma relação entre tarefas e seus proprietários no momento de registro das tarefas na base de dados.

Após o estabelecimento desses pontos, pôde-se ter uma ideia melhor do funcionamento e do fluxo de uso da aplicação. Então, passou-se para a etapa de elaboração da interface gráfica da aplicação. Nessa fase, criou-se as páginas relacionadas ao cadastro e autenticação de usuários. Em seguida, projetou-se uma página inicial para aplicação, uma página para configurar uma nova tarefa, uma página para exibir o progresso da tarefa e uma página para apresentar os resultados da tarefa. Com todas essas seções criadas, pôde-se passar para a próxima fase.

Na fase de implementação e integração da parte lógica, foi preciso empregar as ferramentas comentadas na [seção 3.4](#) para viabilizar a real execução dos otimizadores selecionados na resolução dos problemas representados pelas funções de teste. Na sequência, passou-se para os testes finais. Com a parte lógica pronta e integrada, procurou-se validar os campos e averiguar os resultados apresentados para que eventuais correções pudessem ser feitas. É importante dizer que essa etapa foi chamada de testes finais pois a interface e a lógica foram testadas também durante o decorrer das fases de criação e integração.

A execução desses passos é explanada em detalhes no [Capítulo 4](#), que evidencia as decisões tomadas para concretizar o presente plano de desenvolvimento. Ademais, faz-se necessário esclarecer também as ferramentas empregadas na construção do aplicativo. Logo, a [seção 3.4](#) comenta de forma sucinta os equipamentos e os programas utilizados.

3.4 Ferramentas

Para o desenvolvimento da aplicação, fez-se uso de um computador pessoal com um processador Intel i5-6200U de 2.30 GHz de velocidade, disco rígido de 1 TB e 8 GB de memória principal. É importante dizer que a aplicação não exige muitos recursos, podendo ser executada em máquinas com especificações inferiores. Outrossim, em termos de software, utilizou-se o sistema operacional Ubuntu 22.04.1 de 64 bits, o editor de código Visual Studio Code 1.74.2, o sistema de controle de versões Git, o serviço de hospedagem de código GitHub, a linguagem de programação Python 3.10.6 e outros módulos relacionados ao visual e às funcionalidades.

Pode-se dizer que a escolha da linguagem foi motivada por sua versatilidade. Há bibliotecas para diversas finalidades em Python, sendo uma linguagem bem popular em áreas como otimização e aprendizado de máquina no período vigente. Ao mesmo tempo, Python também conta com *frameworks* que permitem a criação de aplicações para a internet.

No que se refere ao Git¹, empregou-se essa ferramenta para o controle de versões do projeto. Com essa gerência, torna-se possível retornar a estados prévios da base de código bem como criar novos estados pela adição incremental de trechos de código. Por sua vez, o GitHub² integra-se ao Git e permite a hospedagem do código em um servidor remoto para que seja viável não só o compartilhamento do projeto mas também desenvolvimento em conjunto.

Em relação às demais tecnologias utilizadas, pode-se verificar o motivo de escolha bem como uma breve explicação nas subseções que se encontram a seguir.

3.4.1 Opytimizer

[Rosa, Rodrigues e Papa \(2019\)](#) propuseram a Opytimizer, uma biblioteca em Python com a implementação de diversas meta-heurísticas de otimização, visando facilitar a vida de pesquisadores das áreas de pesquisa operacional, aprendizado de máquina, engenharia etc. O objetivo dos autores foi trazer a implementação das heurísticas mais populares e permitir que o uso e a integração com outras aplicações fossem facilitados. Por esse motivo, a Opytimizer, em sua versão 3.1.2, foi utilizada para implementar as tarefas de otimização da aplicação proposta.

¹ Disponível em: <<https://git-scm.com/>>. Acesso em: 25 de jan. 2023.

² Disponível em: <<https://github.com/>>. Acesso em: 25 de jan. 2023.

3.4.2 Opytimark

Ademais, [Rosa, Rodrigues e Papa \(2019\)](#) também propuseram um módulo em Python chamado Opytimark. Essa biblioteca, por sua vez, tem como propósito trazer um conjunto de funções de teste para avaliar a performance de otimizadores meta-heurísticos. Os autores predefiniram essas funções matemáticas como classes da linguagem Python, permitindo que elas possam ser facilmente integradas com outras bibliotecas, como a própria Opytimizer. Logo, a Opytimark 1.0.8 foi empregada para a obtenção dos problemas mencionados na [seção 3.2](#).

3.4.3 Django

De acordo com o DJANGO PROJECT (2022), Django 4.1.2 é um *framework* web de código aberto e de alto nível que utiliza Python para a construção de aplicações para a internet. Essa ferramenta implementa uma versão modificada do padrão MVC, organizando o projeto em modelos, visões e *templates*. Nesse padrão, o Django assume o papel do controlador. Ademais, a visão deixa de formatar os dados e passa o papel para os *templates*. Destarte, levou-se em conta a compatibilidade com as bibliotecas, que usam Python, as facilidades garantidas por um *framework* e o padrão de arquitetura bem estabelecido para a escolha do Django.

Na visão de [Casteleyn et al. \(2009\)](#), um *framework* é uma ferramenta que simplifica o trabalho de desenvolver um aplicativo web do zero, tomando conta de questões como o roteamento das páginas, o gerenciamento de sessões, a formatação de *templates* etc. Em síntese, trata-se de um conjunto de bibliotecas e ferramentas que permitem acelerar o desenvolvimento. Vale mencionar que os *templates* são arquivos escritos em linguagens tradicionais como HTML, CSS e JS, e que possuem como objetivo a estruturação e estilização de páginas, podendo contar com variáveis preenchidas dinamicamente através dos dados fornecidos pelas visões.

3.4.4 Celery e Redis

Segundo [CELERY \(2023\)](#), o Celery 5.2.7 implementa uma fila distribuída de tarefas. O autor explica que esse mecanismo é capaz de adicionar tarefas a uma fila para que processadores ou núcleos de processamento possam executá-las conforme suas disponibilidades. Nesse sentido, uma máquina com muitos núcleos de processamento pode executar tarefas em paralelo, agilizando o trabalho. Em suma, um agente observa a fila e os núcleos da máquina, fazendo a alocação das tarefas assim que unidades de processamento estejam livres.

Nesse esquema, um cliente envia as tarefas que deseja executar ao agente do Celery, trocando mensagens para averiguar o progresso ou solicitar os resultados da execução da tarefa. No caso do presente projeto, o cliente é o Django e o meio escolhido para a troca de informações com o agente é o Redis 4.3.4, um servidor de mensagens bem popular.

De acordo com [Redis \(2023\)](#), o Redis faz a mediação entre o cliente e o agente de um serviço como o Celery. Logo, quando o Django deseja iniciar uma tarefa, adiciona uma

mensagem à fila do Redis, que, por sua vez, entrega a mensagem ao Celery logo que possível. De maneira similar, quando o agente do Celery deseja enviar o resultado ou progresso da tarefa ao Django, ele deposita uma mensagem na fila mantida pelo Redis.

É interessante notar que, como a execução da tarefa ocorre de forma assíncrona, o utilizador da aplicação Django coloca uma ou mais tarefas para executar e pode continuar utilizando funcionalidades do aplicativo. No entanto, para tornar essa lógica viável no caso deste trabalho, foi preciso complementar o Celery com duas extensões. A primeira, Celery Progress 0.1.2³, implementa uma barra de progresso que recebe informações do agente do Celery continuamente, visando apresentar ao usuário o quanto de suas tarefas já foi executado. Por sua vez, a segunda, Django Celery Results 2.4.0⁴, registra as informações da tarefa na base de dados criada pelo Django, garantindo o acesso futuro de detalhes e resultados da execução.

3.4.5 MathJax

O MathJax 3.0⁵ é uma ferramenta de código aberto que permite a renderização de expressões matemáticas escritas por meio da sintaxe do LaTeX, um sistema de formatação de arquivos muito empregado no meio científico. De modo geral, esse módulo JS procura facilitar a apresentação de fórmulas em páginas web não só formatando-as, mas também permitindo que o usuário aumente ou diminua o tamanho da expressão renderizada, realize a cópia da sintaxe da expressão e acione demais opções disponíveis em seu menu de contexto.

3.4.6 Plotly

Em conformidade com Plotly (2023), Plotly 5.11.0 é uma biblioteca em Python para a criação de gráficos interativos de diversos tipos, tais como gráficos de dispersão e de barras. Ademais, essa ferramenta permite a representação do erro dos dados. Na aplicação, utilizou-se o módulo para produzir gráficos com HTML, CSS e JS e inseri-los em inseridos em *templates*.

3.4.7 Bootstrap

Por fim, para a estruturação e estilização dos *templates*, empregou-se a versão 5.2.2 do Bootstrap. Segundo Bootstrap (2023), trata-se de uma ferramenta gratuita que agiliza o desenvolvimento com Linguagem de Marcação de Hipertexto, ou HyperText Markup Language (HTML) no original; Folha de Estilos em Cascata, ou Cascading Style Sheets (CSS); e JavaScript (JS). Essa biblioteca traz estilos predefinidos para as estruturas que compõem as páginas da aplicação, permitindo a criação de interfaces completas e responsivas. Logo, no presente trabalho, Bootstrap auxiliou na rápida construção dos *templates* do Django.

³ Disponível em: <<https://github.com/czue/celery-progress>>. Acesso em: 02 de jan. 2023.

⁴ Disponível em: <<https://django-celery-results.readthedocs.io/en/latest/>>. Acesso em: 02 de jan. 2023.

⁵ Disponível em: <<https://docs.mathjax.org/en/latest/>>. Acesso em: 02 de jan. 2023.

3.5 Métricas

O resultado de uma tarefa de otimização é a melhor solução obtida com o número de iterações definido. Além da melhor solução, pode-se apresentar também o valor da função quando esse vetor de solução é aplicado. Outrossim, quando se executa a tarefa várias vezes, pode-se apresentar a melhor solução dentre todas as execuções, o melhor valor da função, o pior valor da função, o valor médio da função e seu respectivo desvio padrão.

Adicionalmente, [Yang \(2014\)](#) afirma que a convergência, uma medida para quão rápido o otimizador se aproxima de uma solução, pode ser visualizada por meio de um gráfico que acompanhe os valores de função obtidos pelos melhores agentes através das iterações. No caso de múltiplas execuções, pode-se apresentar a média dos melhores valores de função e representar os valores de desvio padrão por meio de uma área sombreada ao redor da curva.

[Yang \(2014\)](#) explica que executar os otimizadores várias vezes garante uma visão mais confiável da convergência e dos resultados obtidos, uma vez que se obtém uma noção da performance média do algoritmo na resolução do problema. Ademais, o desvio padrão ajuda a entender a variabilidade dos valores obtidos, complementando a interpretação da média.

4 Desenvolvimento

Neste capítulo, detalha-se as etapas de desenvolvimento mencionadas no [Capítulo 3](#). Em um primeiro momento, deliberou-se sobre as opções de execução que o usuário poderia configurar, os dados de execução que deveriam ser salvos e os resultados que deveriam ser apresentados ao usuário. Nessa etapa, criou-se a base de dados da aplicação por meio dos modelos e adicionou-se registros por painel administrativo. Na sequência, desenvolveu-se a parte visual com os *templates* do Django, os estilos do Bootstrap e algumas personalizações.

Por último, criou-se a parte lógica da aplicação por meio da junção de módulos externos. Concomitante, integrou-se, aos poucos, a lógica com a interface tendo em vista a obtenção da aplicação final. É importante mencionar que, ao término das atividades, disponibilizou-se a base de código do projeto MetaOPT por meio de um repositório público no GitHub¹.

Na seção [seção 4.1](#), comenta-se as páginas do painel administrativo gerado pelo Django e as tabelas criadas para a base de dados do projeto. Logo em seguida, na [seção 4.2](#), comenta-se a aplicação, explanando tanto a interface quanto a lógica de negócios.

4.1 Painel Administrativo

Uma das funcionalidades do *framework* Django é a geração automática de um site administrativo que é capaz de capturar os modelos ou tabelas da base de dados e permitir a gerência interna do conteúdo da aplicação. O Django suporta muitos tipos de banco de dados, porém sua escolha padrão é o SQLite, uma opção mais leve. O presente projeto optou pelo padrão, fazendo uso do SQLite para criar a base de dados. Sendo assim, logo no início do projeto, executou-se um comando para criar a base juntamente com a tabela de usuários. Então, registrou-se um usuário administrador para o acesso ao painel.

Destarte, as principais páginas desse portal, com os registros das tabelas, são apresentadas abaixo, começando pela página de login da administração na [subseção 4.1.1](#).

4.1.1 Login

A página de login do portal tem como propósito restringir o acesso aos administradores, que são usuários encarregados pela gerência da aplicação. Com usuário e senha em mãos, um administrador pode entrar no portal por meio da página exibida na [Figura 1](#).

¹ Disponível em: <<https://github.com/recogna-lab/metaopt>>. Acesso em: 25 de jan. 2023.

Figura 1 – Login para Administradores.



Fonte: Elaborada pelo autor.

4.1.2 Página Inicial

A [Figura 2](#) apresenta a página inicial da administração do site, que lista todas as tabelas criadas para o projeto. Com exceção da tabela de usuário, que é criada automaticamente no começo do projeto, as demais foram definidas por meio de modelos, conceito já discutido no padrão MVC. Dentre essas tabelas, tem-se que as mais relevantes para o módulo de otimização do MetaOPT são: User, Optimizer, Function, Task Result e User Task.

A tabela User armazena as informações do administrador e dos demais utilizadores da aplicação. Por sua vez, a tabela Function armazena informações das funções de teste escolhidas. A tabela Optimizer guarda os dados referentes aos otimizadores meta-heurísticos selecionados. Já o modelo Task Result fica encarregado dos dados de execução e dos resultados das tarefas. Por fim, o modelo User Task associa os usuários às suas respectivas tarefas.

Figura 2 – Página Inicial da Administração.



Fonte: Elaborada pelo autor.

4.1.2.1 Lista de Usuários

Na [Figura 3](#), pode-se observar a listagem apresentada ao clicar na opção **Usuários** da página inicial supracitada. Trata-se de todos os usuários cadastrados no sistema. Nessa lista, apresenta-se os principais campos referentes aos utilizadores, tais como identificador, nome de usuário, e-mail, primeiro e último nome, e um campo que indica se o usuário é ou não um membro da equipe de administração do aplicativo. Ademais, no canto superior esquerdo da imagem, nota-se a presença de um campo de busca que permite encontrar usuários por nome de usuário, sobrenome, e-mail etc. Outrossim, no canto superior direito, pode-se notar a presença de um botão para criação de novas entradas na tabela User.

Figura 3 – Usuários Cadastrados.

Selecione usuário para modificar ADICIONAR USUÁRIO +

Q Pesquisar

Ação: Ir 0 de 2 selecionados

<input type="checkbox"/>	ID	USUÁRIO	ENDEREÇO DE EMAIL	PRIMEIRO NOME	ÚLTIMO NOME	MEMBRO DA EQUIPE
<input type="checkbox"/>	1	admin	admin@gmail.com			✓
<input type="checkbox"/>	3	usuario.tcc	usuario@gmail.com	usuario	fulano	✗

2 usuários

Fonte: Elaborada pelo autor.

4.1.2.2 Lista de Otimizadores

De modo análogo, pode-se ver, na [Figura 4](#), os registros presentes na tabela de otimizador, acessível pela opção **Optimizers**. Trata-se de todas as meta-heurísticas cadastradas no aplicativo via painel administrativo. Nessa listagem, apresenta-se os campos nome e acrônimo das técnicas. Ademais, nota-se novamente o botão para a inserção de novos registros.

Figura 4 – Otimizadores Cadastrados.

Selecione optimizer para modificar ADICIONAR OPTIMIZER +

Ação: Ir 0 de 6 selecionados

<input type="checkbox"/>	OPTIMIZER
<input type="checkbox"/>	Algoritmo Genético (GA)
<input type="checkbox"/>	Algoritmo do Vaga-lume (FA)
<input type="checkbox"/>	Busca Cuco (CS)
<input type="checkbox"/>	Colônia Artificial de Abelhas (ABC)
<input type="checkbox"/>	Otimização por Enxame de Partículas (PSO)
<input type="checkbox"/>	Recozimento Simulado (SA)

6 optimizers

Fonte: Elaborada pelo autor.

4.1.2.3 Lista de Funções de Teste

A [Figura 5](#) apresenta as entradas cadastradas na tabela Function, acessível pela opção **Functions** da página de administração. Os campos presentes na imagem correspondem ao nome da função, sua expressão em LaTeX, os limites de seu domínio e, por fim, o ótimo global. De modo semelhante, pode-se notar o botão para a criação de novas entradas.

Figura 5 – Funções de Teste Cadastradas.

Selecione function para modificar ADICIONAR FUNCTION +

Ação: 0 de 10 selecionados

<input type="checkbox"/>	NAME	LATEX EXPRESSION	BOUND	OPTIMAL RESULT
<input type="checkbox"/>	Função Ackley 1	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi x_i)} + 20 + e$	{ "lower": -32, "upper": 32 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Alpine 1	$f(\mathbf{x}) = \sum_{i=1}^n x_i \sin(x_i) + 0.1 x_i $	{ "lower": -10, "upper": 10 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Csendes	$f(\mathbf{x}) = \sum_{i=1}^n x_i^6 (2 + \sin(\frac{1}{x_i}))$	{ "lower": -1, "upper": 1 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Griewank	$f(\mathbf{x}) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}})$	{ "lower": -100, "upper": 100 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Powell Sum	$f(\mathbf{x}) = \sum_{i=1}^n x_i ^{i+1}$	{ "lower": -1, "upper": 1 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Rastrigin	$f(\mathbf{x}) = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i))$	{ "lower": -5.12, "upper": 5.12 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	{ "lower": -30, "upper": 30 }	{ "solution": 1, "value": 0 }
<input type="checkbox"/>	Função Salomon	$f(\mathbf{x}) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^n x_i^2}) + 0.1 \sqrt{\sum_{i=1}^n x_i^2}$	{ "lower": -100, "upper": 100 }	{ "solution": 0, "value": 0 }
<input type="checkbox"/>	Função Schwefel	$f(\mathbf{x}) = 418.9829n - \sum_{i=1}^n x_i \sin(\sqrt{ x_i })$	{ "lower": -100, "upper": 100 }	{ "solution": 420.9687, "value": 0 }
<input type="checkbox"/>	Função Step	$f(\mathbf{x}) = \sum_{i=1}^n x_i $	{ "lower": -100, "upper": 100 }	{ "solution": 0, "value": 0 }

10 functions

Fonte: Elaborada pelo autor.

4.1.2.4 Lista de Tarefas

Por sua vez, na [Figura 6](#), lista-se os registros cadastrados na tabela Task Result, acessível pela opção **Task results** do painel. As informações presentes na imagem correspondem aos principais dados da tarefa, de sua execução e dos resultados gerados. Em síntese, optou-se por armazenar um identificador, o estado, o tipo da tarefa, os argumentos passados na criação da tarefa, os resultados produzidos pela tarefa, a data e hora de criação, e a data e hora de finalização. Esses campos são importantes para a consulta posterior das tarefas, de seus resultados e de seus gráficos de convergência. Diferente das demais listagens, nota-se que não há botão para adição, pois as entradas devem ser produzidas pela lógica da aplicação.

Figura 6 – Tarefas Cadastradas.

Selecione task result para visualizar

2023 24 de Janeiro

Ação: 0 de 4 selecionados

<input type="checkbox"/>	TASK ID	TASK STATE	TASK NAME	TASK ARGUMENTS	RESULT DATA	CREATED DATETIME	COMPLETED DATETIME
<input type="checkbox"/>	7a5e84ef-e242-4f07-9d7c-64d8806324ef	-	optimization	{ "user_id": 1, "optimizer": "CS", "function": "Rastrigin", "dimension": 30, "b...	{ "pending": false, "current": 66, "total": 500, "percent": 13.2, "description": ...	24 de Janeiro de 2023 às 19:51	24 de Janeiro de 2023 às 19:53
<input type="checkbox"/>	3e8dac2b-9fbf-40ff-b097-1c92436ac1f0	REVOKED	optimization	{ "user_id": 1, "optimizer": "PSO", "function": "Alpine1", "dimension": 10, "bo...	{ "exc_type": "TaskRevokedError", "exc_message": ["terminated"], "exc_module": ...	24 de Janeiro de 2023 às 19:49	24 de Janeiro de 2023 às 19:49
<input type="checkbox"/>	046574ed-2aa9-4441-917a-3519a9be189b	SUCCESS	optimization	{ "user_id": 1, "optimizer": "ABC", "function": "Rosenbrock", "dimension": 20, ...	{ "best_solution": [-1.4429062514584874, -0.6972694414155357, -2.25825268986471...	24 de Janeiro de 2023 às 19:47	24 de Janeiro de 2023 às 19:47
<input type="checkbox"/>	018df5b3-1c90-43b5-86ac-1c2795ffa094	SUCCESS	optimization	{ "user_id": 1, "optimizer": "GA", "function": "PowellSum", "dimension": 30, "b...	{ "best_solution": [0.006450739671500606, -0.05592282007462546, 0.0069865968896...	24 de Janeiro de 2023 às 19:43	24 de Janeiro de 2023 às 19:45

4 task results

Fonte: Elaborada pelo autor.

4.1.2.5 Lista de Tarefas dos Usuários

Por último, na figura [Figura 7](#), lista-se os registros cadastrados na tabela User Task, acessível pela opção **User tasks** do painel. Essa tabela serve para a associação dos usuários às suas tarefas. Em suma, armazena-se somente dois identificadores, um do usuário e o outro da tarefa. No entanto, por meio desses identificadores, pode-se carregar os campos apresentados na imagem, que são informações importantes do usuário e da tarefa. Como as entradas dessa tabela são registradas automaticamente quando uma tarefa é inserida na tabela Task Result, nota-se que não há botão para a inserção de novas entradas.

Figura 7 – Associação das Tarefas aos seus Proprietários.

Selecione user task para visualizar

2023
24 de Janeiro

Ação:
Ir
0 de 4 selecionados

<input type="checkbox"/>	USERNAME	EMAIL	TASK ID	TASK STATE	TASK NAME	CREATED DATETIME	COMPLETED DATETIME
<input type="checkbox"/>	admin	admin@gmail.com	7a5e84ef-e242-4f07-9d7c-64d8806324ef	PROGRESS	optimization	24 de Janeiro de 2023 às 19:51	24 de Janeiro de 2023 às 19:56
<input type="checkbox"/>	admin	admin@gmail.com	3e8dac2b-9fbf-40ff-b097-1c92436ac1f0	REVOKED	optimization	24 de Janeiro de 2023 às 19:49	24 de Janeiro de 2023 às 19:49
<input type="checkbox"/>	admin	admin@gmail.com	046574ed-2aa9-4441-917a-3519a9be189b	SUCCESS	optimization	24 de Janeiro de 2023 às 19:47	24 de Janeiro de 2023 às 19:47
<input type="checkbox"/>	admin	admin@gmail.com	018df5b3-1c90-43b5-86ac-1c2795ffa094	SUCCESS	optimization	24 de Janeiro de 2023 às 19:43	24 de Janeiro de 2023 às 19:45

4 user tasks

Fonte: Elaborada pelo autor.

4.2 MetaOPT

Nesta seção, apresenta-se as principais páginas do projeto MetaOPT, levando em conta o escopo do módulo de otimização. Além disso, busca-se explicitar o fluxo de uso da aplicação.

4.2.1 Login

A Figura 8 ilustra a página de login na qual os usuários da aplicação devem entrar com usuário e senha para obterem acesso à página, denominada *dashboard*. Nessa imagem, pode-se observar dois botões, um para acessar à *dashboard* e outro para exibir a senha que, por padrão, tem os caracteres substituídos por asteriscos. Ademais, há dois links no formulário. O primeiro leva o usuário à página de redefinição de senha para que ele recupere o acesso à sua conta. Já o segundo leva o usuário para o cadastro para que ele possa criar uma conta.

É importante notar a presença do menu superior e do rodapé, que estão presentes em todas as páginas e contam com a identidade visual do projeto. No menu, tem-se o nome do projeto à esquerda e links de navegação à direita. O nome leva à *dashboard* quando o usuário está logado, porém vale informar que qualquer tentativa de acesso às páginas restritas por parte de visitantes não identificados redireciona à página de login. Por sua vez, os links resumem as páginas que um usuário não identificado tem acesso, como login, cadastro e redefinição de senha. Na segunda parte do menu, há uma descrição da página atual à esquerda e uma mensagem de bem-vindo à direita. Por fim, no rodapé, há uma breve descrição do projeto.

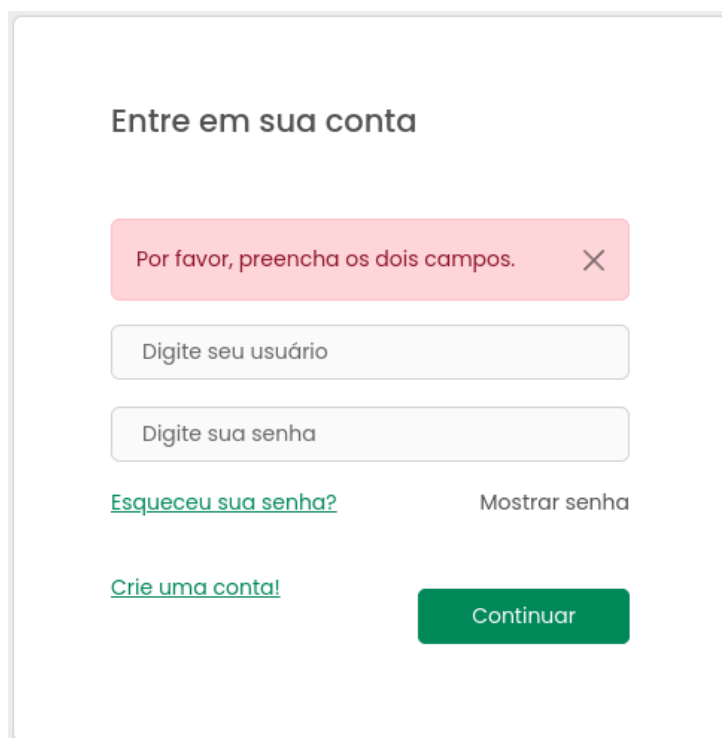
Figura 8 – Login para Usuários Comuns.

The image shows a web application interface for MetaOPT. At the top, there is a blue header bar with the text 'MetaOPT' on the left and navigation links 'Login', 'Cadastro', and 'Redefinição de Senha' on the right. Below the header, a grey bar contains the word 'Login' on the left and 'Bem-vindo, visitante!' on the right. The main content area is light grey and features a white login form in the center. The form is titled 'Entre em sua conta' and contains two input fields: 'Digite seu usuário' and 'Digite sua senha'. Below the password field, there are two links: 'Esqueceu sua senha?' and 'Mostrar senha'. At the bottom of the form, there is a link 'Crie uma conta!' and a green button labeled 'Continuar'. The footer of the page is a blue bar with the text 'MetaOPT - Meta-Heurísticas em Tarefas de Otimização e de Seleção de Características'.

Fonte: Elaborada pelo autor.

Na [Figura 9](#), exibe-se a mensagem de erro que o usuário recebe quando tenta entrar na *dashboard* sem ter preenchido os dois campos do formulário. Como esses campos são necessários para o login, pede-se que o usuário os preencha para poder entrar. Caso uma das informações esteja inválida, também é preciso notificar o usuário. Logo, a [Figura 10](#) exibe a mensagem de erro apresentada quando o usuário ou a senha estão inválidos.


Figura 9 – Login com Erro de Campos Vazios.



The image shows a login form titled "Entre em sua conta". At the top, there is a red error message box that says "Por favor, preencha os dois campos." with a close button (X). Below this, there are two input fields: "Digite seu usuário" and "Digite sua senha". Under the password field, there is a link "Esqueceu sua senha?" and a toggle "Mostrar senha". At the bottom left, there is a link "Crie uma conta!". At the bottom right, there is a green "Continuar" button.

Fonte: Elaborada pelo autor.

Figura 10 – Login com Erro de Dados Inválidos.



The image shows a login form titled "Entre em sua conta". At the top, there is a red error message box that says "Usuário ou senha inválidos." with a close button (X). Below this, there are two input fields: "usuario123" and "12345". Under the password field, there is a link "Esqueceu sua senha?" and a toggle "Esconder senha". At the bottom left, there is a link "Crie uma conta!". At the bottom right, there is a green "Continuar" button.

Fonte: Elaborada pelo autor.

4.2.2 Cadastro

Dando continuidade, a [Figura 11](#) mostra a seção de cadastro na qual usuários que não possuem conta podem se registrar preenchendo os campos nome, sobrenome, usuário, e-mail, senha e confirmação de senha. A confirmação de senha, diferente dos demais campos, não é armazenada na base de dados, servindo apenas como uma garantia de que o usuário conhece bem a senha que digitou. Na página, pode-se notar que há dois botões, um para exibir o conteúdo dos campos de senha e outro para efetuar o cadastro. Além do mais, há um link de navegação no formulário para que o usuário possa retornar à página de login. Outrossim, é possível observar que, como a página de cadastro não é central como a página de login ou a de *dashboard*, ao lado da descrição da página do menu superior surge um link de retornar para que o usuário possa facilmente regressar à página de login do sistema.

Figura 11 – Cadastro de Usuários Comuns.

MetaOPT

Login Cadastro Redefinição de Senha

Cadastro | Retornar Bem-vindo, visitante!

Crie uma conta

Digite seu nome
 Digite seu sobrenome

Digite seu usuário

Digite seu e-mail

Digite sua senha
 Repita sua senha

Guarde bem suas informações cadastrais.
 [Mostrar senha](#)

[Entre em sua conta!](#)

MetaOPT - Meta-Heurísticas em Tarefas de Otimização e de Seleção de Características

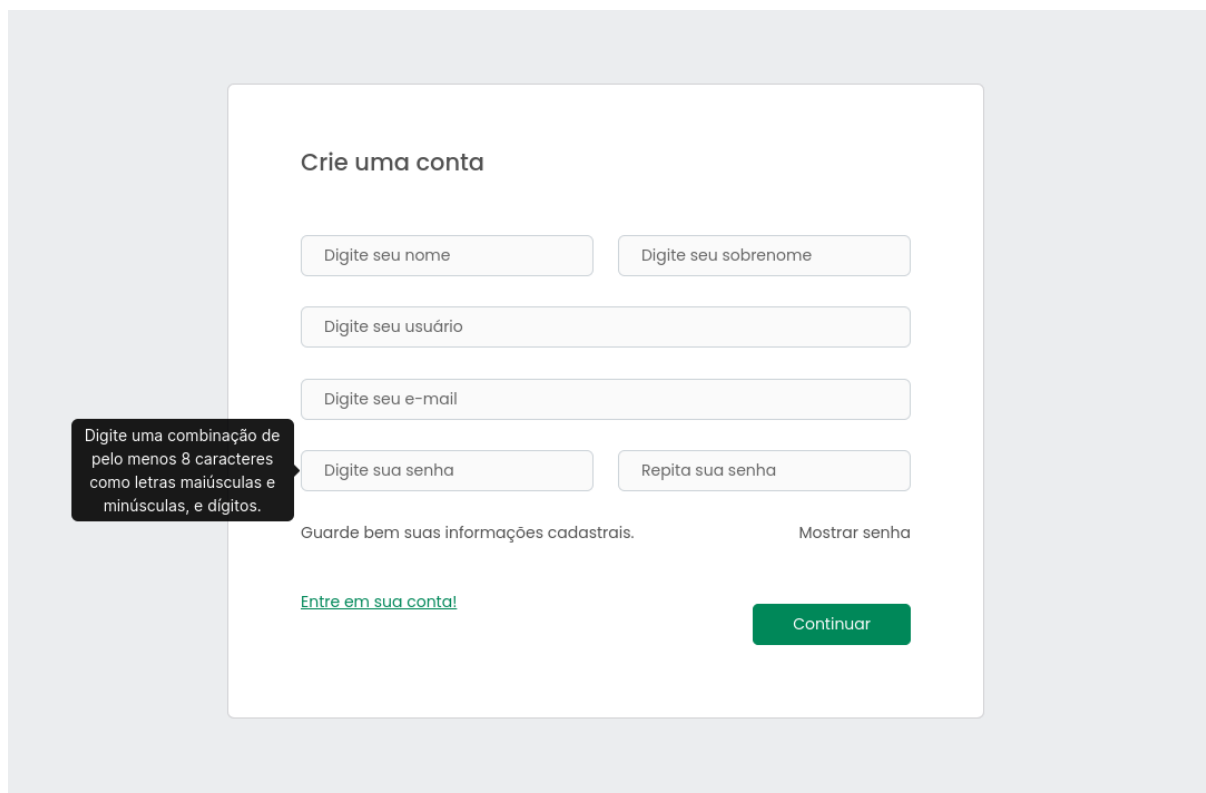
Fonte: Elaborada pelo autor.

Ainda no cadastro, há um balão de dica, denominado *tooltip*, que informa ao usuário como ele deve preencher cada um dos campos do formulário. Esse balão torna-se visível quando o usuário passa o mouse sobre um campo ou começa a digitar nele, como ilustrado na [Figura 12](#). Ademais, em relação aos erros dessa seção, a [Figura 13](#) exibe o erro de campos vazios, que lembra o usuário de preencher todos os campos antes de efetuar o cadastro. Em seguida, a [Figura 14](#) lembra o utilizador que o campo usuário precisa ser válido. Importante dizer que esse mesmo erro de invalidade pode ocorrer com o e-mail ou a senha.

Para o campo usuário, deve-se informar um nome que possua dígitos ou letras e somente alguns caracteres especiais, que o usuário confere pelo *tooltip* do campo. Além disso, o nome

de usuário não pode coincidir com o nome de outro usuário. Já o e-mail, precisa estar em um formato adequado, com '@' e '.', além de não coincidir com o e-mail de um usuário já cadastrado. Por último, a senha é considerada válida quando possui pelo menos um dígito, uma letra minúscula, uma maiúscula e totalize oito caracteres no mínimo.

Figura 12 – Cadastro com *Tooltip*.



O formulário, intitulado "Crie uma conta", contém os seguintes campos e elementos:

- Dois campos de texto para "Nome" e "Sobrenome".
- Um campo de texto para "Usuário".
- Um campo de texto para "E-mail".
- Dois campos de texto para "Senha" e "Repita sua senha".
- Um link "Mostrar senha" próximo ao campo de repetição da senha.
- Um link "Entre em sua conta!" na base do formulário.
- Um botão verde "Continuar" na base direita.

Um tooltip preto com texto branco aponta para o campo de senha, contendo a seguinte mensagem: "Digite uma combinação de pelo menos 8 caracteres como letras maiúsculas e minúsculas, e dígitos."

Fonte: Elaborada pelo autor.

Figura 13 – Cadastro com Erro de Campos Vazios.

Crie uma conta

Por favor, preencha todos os campos corretamente. X

Digite seu nome

Por favor, digite seu nome.

Digite seu sobrenome

Por favor, digite seu sobrenome.

Digite seu usuário

Por favor, digite seu usuário.

Digite seu e-mail

Por favor, digite seu e-mail.

Digite sua senha

Por favor, digite sua senha.

Repita sua senha

Por favor, repita sua senha.

Guarde bem suas informações cadastrais.

Mostrar senha

[Entre em sua conta!](#)

Continuar

Fonte: Elaborada pelo autor.

Figura 14 – Cadastro com Erro de Usuário Inválido.

Crie uma conta

Por favor, preencha todos os campos corretamente. X

Fulano ✓
Parece certo!

De tal ✓
Parece certo!

usuario/tcc ⓘ
Informe um nome de usuário válido. Este valor pode conter apenas letras, números e os seguintes caracteres @/./+/-/_.

usuario.tcc@gmail.com ✓
Parece certo!

Digite sua senha ✓
Parece certo!

Repita sua senha ✓
Parece certo!

Guarde bem suas informações cadastrais.

Mostrar senha

[Entre em sua conta!](#)

Continuar

Fonte: Elaborada pelo autor.

Ainda na parte de cadastro, pode ocorrer o erro de senhas distintas, como demonstrado na [Figura 15](#). Esse erro ocorre quando o campo de confirmação de senha não possui o mesmo conteúdo do campo senha, o que indica uma incoerência por parte do usuário. Entretanto, caso todos os campos estejam preenchidos, sejam válidos e as senhas combinem, o usuário pode concluir seu cadastro sem empecilhos. Ao finalizar o registro, o usuário é redirecionado ao login onde uma mensagem de sucesso é exibida como na [Figura 16](#).

Figura 15 – Cadastro com Erro de Senhas Distintas.

Crie uma conta

Por favor, preencha todos os campos corretamente. X

Fulano ✓
Parece certo!

De tal ✓
Parece certo!

usuario.tcc ✓
Parece certo!

usuario.tcc@gmail.com ✓
Parece certo!

Tcc12345 ✓
Parece certo!

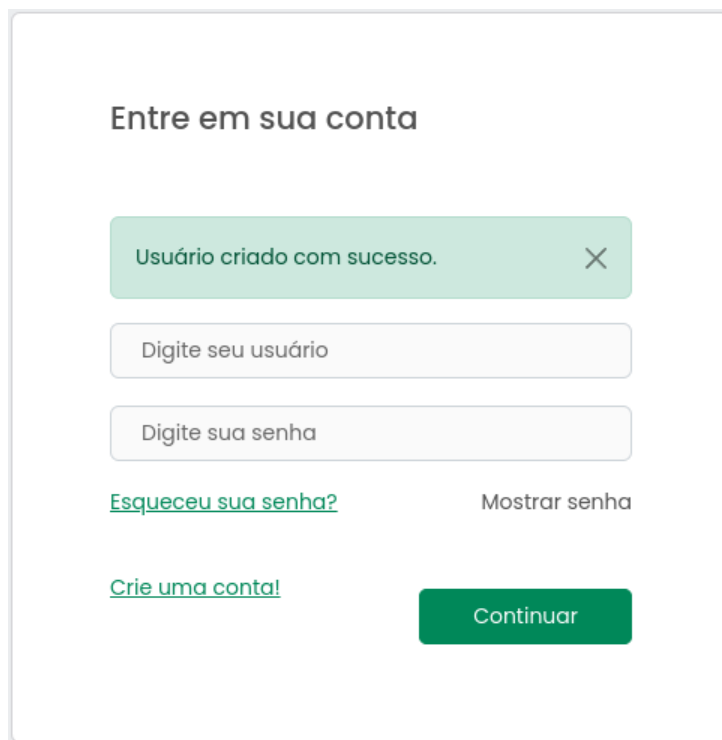
Tcc12235 ⓘ
O campo "Confirmar senha" deve ser igual ao "Senha".

Guarde bem suas informações cadastrais. Esconder senha

[Entre em sua conta!](#) Continuar

Fonte: Elaborada pelo autor.

Figura 16 – Login com Mensagem de Usuário Criado.



The image shows a login interface titled "Entre em sua conta". At the top, there is a green success message box that says "Usuário criado com sucesso." with a close button (X). Below this are two input fields: "Digite seu usuário" and "Digite sua senha". Under the password field, there is a link "Esqueceu sua senha?" and a text "Mostrar senha". At the bottom left, there is a link "Crie uma conta!". At the bottom right, there is a green button labeled "Continuar".

Fonte: Elaborada pelo autor.

4.2.3 Redefinição de Senha

A última seção do sistema disponível aos usuários não identificados é a redefinição de senha, representada na [Figura 17](#). Essa funcionalidade permite que um usuário defina uma nova senha, caso tenha perdido a senha definida previamente. Para isso, o usuário digita seu e-mail e efetua o pedido de redefinição. Caso o e-mail digitado não seja encontrado, uma mensagem de erro é exibida ao usuário, conforme indicado pela [Figura 18](#). Nessa página, por simplicidade, não é feita a verificação de campo vazio nem válido, pois campos vazios ou inválidos levam ao erro de e-mail não encontrado, o que está semanticamente correto.

Figura 17 – Redefinição de Senha.

A interface de redefinição de senha do MetaOPT. No topo, uma barra azul contém o logo 'MetaOPT' e os links 'Login', 'Cadastro' e 'Redefinição de Senha'. Abaixo, uma barra cinza mostra 'Redefinição de Senha' em destaque, com um link 'Retornar' e a mensagem 'Bem-vindo, visitante!'. O formulário centralizado, intitulado 'Peça uma nova senha', possui um campo de entrada para o e-mail com o placeholder 'Digite seu-email' e um botão verde 'Continuar'.

Fonte: Elaborada pelo autor.

Figura 18 – Redefinição com Erro de E-mail Não Encontrado.

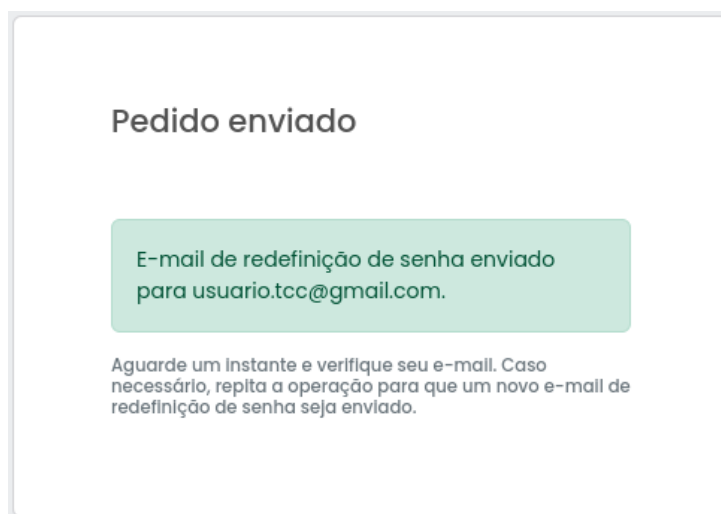
A interface de redefinição de senha do MetaOPT exibindo uma mensagem de erro. O formulário, intitulado 'Peça uma nova senha', apresenta uma caixa de mensagem vermelha com o texto 'E-mail não encontrado.' e um ícone de fechar. Abaixo, o campo de e-mail contém o texto 'usuario.tcc@gmail.com' e o botão verde 'Continuar' permanece visível.

Fonte: Elaborada pelo autor.

Caso o endereço de e-mail digitado exista na base de dados, um e-mail de redefinição de senha é enviado para o endereço e uma mensagem informando o sucesso do envio é apresentada, conforme a [Figura 19](#). Na mesma página, o usuário é avisado para aguardar alguns instantes e verificar sua caixa de e-mail, na qual deve encontrar um e-mail semelhante ao apresentado na

Figura 20. Esse e-mail informa o pedido recebido e recomenda que o usuário ignore a mensagem caso não o tenha feito. No entanto, caso o usuário tenha feito o pedido, ele pode utilizar o link presente no e-mail para acessar a página da **Figura 21**. É importante mencionar que o link presente no e-mail é composto por um identificador de usuário e um *token* que indicava o último acesso do usuário. Dessa forma, é possível saber o usuário que fez o pedido e verificar se esse usuário realizou algum login após a solicitação, fato que invalida o e-mail de alteração de senha já que o acesso indica que o usuário se lembra de sua senha. Se o link estiver inválido, apresenta-se uma tela de erro similar à página apresentada na **subseção 4.2.8**.

Figura 19 – Redefinição com Mensagem de E-mail Enviado.



Fonte: Elaborada pelo autor.

Figura 20 – E-mail de Redefinição Recebido pelo Usuário.



Fonte: Elaborada pelo autor.

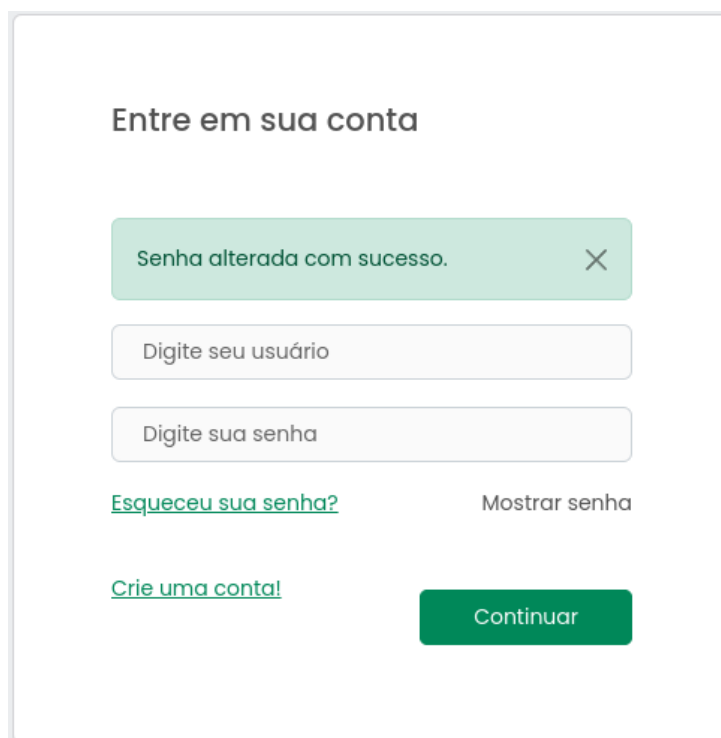
Figura 21 – Alteração de Senha.

The screenshot displays the 'Redefinição de Senha' (Reset Password) page of the MetaOPT application. The page features a blue header with the 'MetaOPT' logo on the left and navigation links for 'Login', 'Cadastro', and 'Redefinição de Senha' on the right. Below the header, a grey bar contains the page title 'Redefinição de Senha' and a welcome message 'Bem-vindo, visitante!'. The main content area is light grey and contains a white card with the heading 'Redefina sua senha'. Inside the card, there are two input fields: 'Digite a nova senha' and 'Repita sua senha'. A 'Mostrar senha' link is positioned below the second input field. A green 'Continuar' button is located at the bottom of the card. The footer of the page is blue and contains the text 'MetaOPT - Meta-Heurísticas em Tarefas de Otimização e de Seleção de Características'.

Fonte: Elaborada pelo autor.

É importante lembrar que a página de alteração de senha também faz a verificação de senha válida e de igualdade das duas senhas. Se as condições forem atendidas, a senha é alterada na base de dados e o usuário é levado à seção de login, na qual se exibe uma mensagem de senha alterada com sucesso como ilustrado pela [Figura 22](#).

Figura 22 – Login com Mensagem de Senha Alterada.



The image shows a login form titled "Entre em sua conta". At the top, there is a green success message box that says "Senha alterada com sucesso." with a close button (X). Below this, there are two input fields: "Digite seu usuário" and "Digite sua senha". To the right of the password field is a link "Mostrar senha". Below the input fields, there are two links: "Esqueceu sua senha?" and "Crie uma conta!". At the bottom right, there is a green button labeled "Continuar".

Fonte: Elaborada pelo autor.

4.2.4 Dashboard

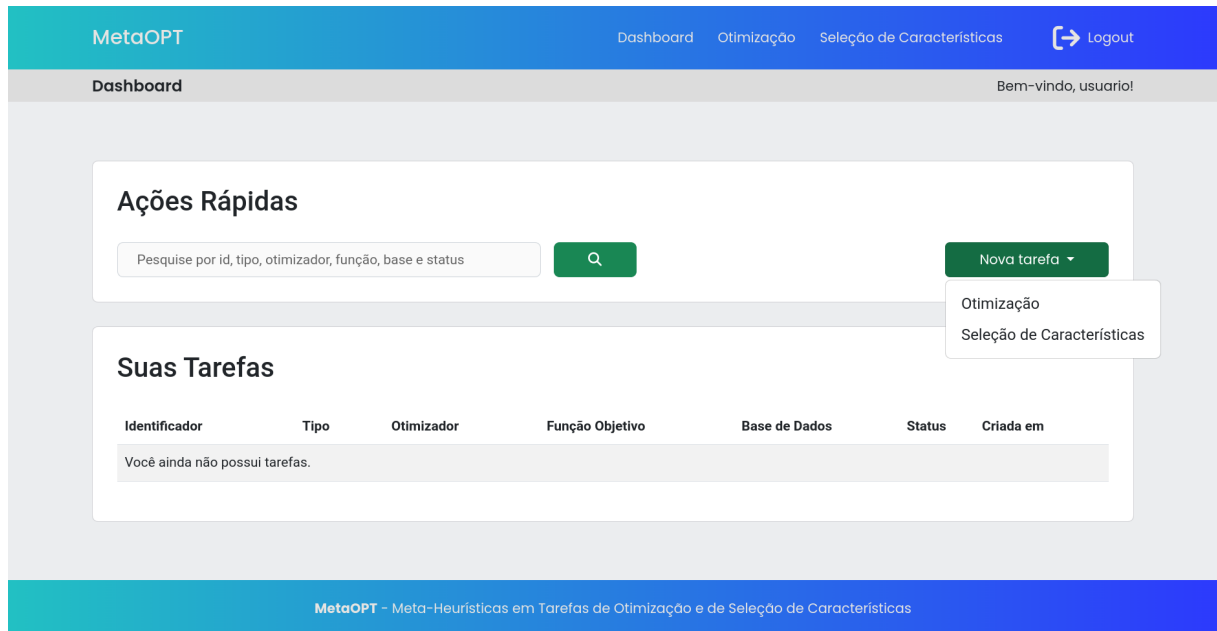
A partir desta subseção, passa-se a comentar as páginas do sistema que são disponibilizadas aos usuários logados. Na [Figura 23](#), apresenta-se a página principal da aplicação, denominada *dashboard*. Nota-se que o menu superior passa por mudanças. Ao lado direito, os links de navegação são atualizados com as páginas internas do aplicativo e a mensagem de bem-vindo recebe o primeiro nome do usuário que realizou o acesso.

Em relação aos links, há uma opção para a própria *dashboard*, uma para página de otimização e uma última para a página de seleção de características. Ademais, no corpo da página, encontra-se uma área superior na qual o usuário pode realizar buscas por tarefas específicas e criar novas tarefas por meio do botão à direita. Vale mencionar que a pesquisa pode ser realizada considerando os campos identificador, tipo, otimizador, função objetivo e estado no caso das tarefas de otimização. O campo de base de dados aplica-se somente às tarefas de seleção de características, que fazem parte do escopo de outro módulo.

Ainda em relação à *dashboard*, observa-se que na segunda área do corpo da página há uma tabela na qual as tarefas são listadas. No caso da [Figura 23](#), nota-se uma tabela sem tarefas registradas. Por outro lado, na [Figura 24](#), pode-se observar uma tabela com quatro tarefas, cada uma com um estado diferente. Dentre os estados possíveis, cita-se Pendente, Progresso, Cancelada e Concluída. Outrossim, nessa mesma imagem, é possível verificar que

o identificador de cada tarefa é um link para a tela de detalhes da tarefa. Adicionalmente, observa-se que o campo base de dados apresenta um indicativo de que não se aplica às tarefas.

Figura 23 – *Dashboard*



Fonte: Elaborada pelo autor.

Figura 24 – Tabela de Tarefas da *Dashboard*

Identificador	Tipo	Otimizador	Função Objetivo	Base de Dados	Status	Criada em
7a5e84ef-e242-4f07-9d7c-64d8806324ef	Otimização	CS	Rastrigin	Não se aplica	Progresso	24 de Janeiro de 2023 às 19:51
3e8dac2b-9fbf-40ff-b097-1c92436ac1f0	Otimização	PSO	Alpine1	Não se aplica	Cancelada	24 de Janeiro de 2023 às 19:49
046574ed-2aa9-4441-917a-3519a9be189b	Otimização	ABC	Rosenbrock	Não se aplica	Pendente	24 de Janeiro de 2023 às 19:47
018df5b3-1c90-43b5-86ac-1c2795ffa094	Otimização	GA	PowellSum	Não se aplica	Sucesso	24 de Janeiro de 2023 às 19:43

Fonte: Elaborada pelo autor.

4.2.5 Nova Tarefa de Otimização

Ao clicar no botão de nova tarefa ou utilizar o link do menu superior, direciona-se para a página de nova tarefa de otimização, que é exibida pela [Figura 25](#). Nessa página, encontra-se um formulário para configuração de novas tarefas. Na parte superior do formulário, há um texto explicativo a respeito dos otimizadores e das funções de teste. O corpo do formulário

conta com as opções de execução da tarefa. Como pode-se observar, o usuário pode escolher o otimizador, a função de teste, o número de dimensões da função, o número de agentes do otimizador, o número de iterações da tarefa e, por fim, o número de execuções ou repetições da tarefa. Na parte inferior do formulário, encontra-se o botão para iniciar a nova tarefa.

É importante mencionar que o número de dimensões, o número de agentes, o número de iterações e o número de execuções possuem valores mínimos e máximos que são apresentados por balões de ajuda. Para o número de dimensões, optou-se por um mínimo de uma e máximo de 30 execuções. Esse máximo foi orientado pelo estudo de [Hussain et al. \(2017\)](#), que indicou 30 dimensões como um valor frequentemente empregado em testes de performance. Em relação ao número de agentes, limitou-se de 5 a 50 agentes. Ademais, limitou-se o número de iterações de 10 a 500. Outrossim, limitou-se o número de execuções de uma a 30, já que o estudo de [Hussain et al. \(2017\)](#) indica 30 execuções como um valor frequente.

Figura 25 – Nova Tarefa de Otimização.

Configure sua tarefa

As tarefas de otimização do MetaOPT empregam meta-heurísticas para minimizar uma função de teste, uma vez que elas são flexíveis e capazes de encontrar boas soluções para vários problemas. Quanto às funções de teste, pode-se dizer que são problemas artificiais de diferentes níveis de dificuldade, servindo para avaliar a performance e a convergência dos otimizadores. Disponibilizam-se aqui os algoritmos e as funções mais populares da literatura.

Otimizador:

Selecione um otimizador

Função de Teste:

Selecione uma função

Dimensão (n):

10

Número de Agentes:

10

Número de Iterações:

50

Número de Execuções:

1

Iniciar tarefa

Fonte: Elaborada pelo autor.

Ainda no formulário de nova tarefa, observa-se que a lista de otimizadores disponíveis é exibida ao selecionar o campo 'Otimizador', como ilustrado na [Figura 26](#). De modo análogo, a lista de funções é exibida ao selecionar o campo 'Função de Teste', como representado pela [Figura 27](#). Nota-se nas imagens que os balões de ajuda também estão presentes para esses campos do formulário. Caso o otimizador e a função não sejam selecionados ou algum valor inválido seja inserido, apresenta-se uma mensagem de erro como na [Figura 28](#).

Figura 26 – Nova Tarefa de Otimização com Otimizadores.

Configure sua tarefa

As tarefas de otimização do MetaOPT empregam meta-heurísticas para minimizar uma função de teste, uma vez que elas são flexíveis e capazes de encontrar boas soluções para vários problemas. Quanto às funções de teste, pode-se dizer que são problemas artificiais de diferentes níveis de dificuldade, servindo para avaliar a performance e a convergência dos otimizadores. Disponibilizam-se aqui os algoritmos e as funções mais populares da literatura.

Otimizador:

Selecione um otimizador

Selecione um otimizador

Algoritmo Genético (GA)

Algoritmo do Vaga-lume (FA)

Busca Cuco (CS)

Colônia Artificial de Abelhas (ABC)

Otimização por Enxame de Partículas (PSO)

Recozimento Simulado (SA)

10

Escolha uma das meta-heurísticas de otimização.

Número de Agentes:

10

Número de Iterações:

50

Número de Execuções:

1

Iniciar tarefa

Fonte: Elaborada pelo autor.

Figura 27 – Nova Tarefa de Otimização com Funções.

Configure sua tarefa

As tarefas de otimização do MetaOPT empregam meta-heurísticas para minimizar uma função de teste, uma vez que elas são flexíveis e capazes de encontrar boas soluções para vários problemas. Quanto às funções de teste, pode-se dizer que são problemas artificiais de diferentes níveis de dificuldade, servindo para avaliar a performance e a convergência dos otimizadores. Disponibilizam-se aqui os algoritmos e as funções mais populares da literatura.

Otimizador:

Selecione um otimizador

Função de Teste:

Selecione uma função

Selecione uma função

Função Ackley 1

Função Alpine 1

Função Csendes

Função Griewank

Função Powell Sum

Função Rastrigin

Função Rosenbrock

Função Salomon

Função Schwefel

Função Step

50

Número de Execuções:

1

Iniciar tarefa

Escolha uma das funções de teste para a tarefa de otimização.

Fonte: Elaborada pelo autor.

Figura 28 – Nova Tarefa de Otimização com Erro.

Configure sua tarefa

As tarefas de otimização do MetaOPT empregam meta-heurísticas para minimizar uma função de teste, uma vez que elas são flexíveis e capazes de encontrar boas soluções para vários problemas. Quanto às funções de teste, pode-se dizer que são problemas artificiais de diferentes níveis de dificuldade, servindo para avaliar a performance e a convergência dos otimizadores. Disponibilizam-se aqui os algoritmos e as funções mais populares da literatura.

Por favor, selecione todas as opções e insira valores válidos. ✕

Otimizador:

Selecione um otimizador

Função de Teste:

Selecione uma função

Dimensão (n):

10

Número de Agentes:

10

Número de Iterações:

50

Número de Execuções:

1

Iniciar tarefa

Fonte: Elaborada pelo autor.

Outro interessante detalhe do formulário de nova tarefa é que, ao selecionar uma função de teste, a expressão de sua forma geral é renderizada logo abaixo do campo de função por meio do MathJax. Essa renderização pode ser conferida na [Figura 29](#), que apresenta um formulário pronto para a execução. Quando o usuário aperta o botão de iniciar, a tarefa é adicionada à fila de tarefas do Celery e o usuário é redirecionado à página de detalhes da tarefa, na qual pode conferir os parâmetros escolhidos para a tarefa e esperar até que ela seja executada.

Figura 29 – Nova Tarefa de Otimização com Função Renderizada.

Configure sua tarefa

As tarefas de otimização do MetaOPT empregam meta-heurísticas para minimizar uma função de teste, uma vez que elas são flexíveis e capazes de encontrar boas soluções para vários problemas. Quanto às funções de teste, pode-se dizer que são problemas artificiais de diferentes níveis de dificuldade, servindo para avaliar a performance e a convergência dos otimizadores. Disponibilizam-se aqui os algoritmos e as funções mais populares da literatura.

Otimizador:

Otimização por Enxame de Partículas (PSO)

Função de Teste:

Função Rosenbrock

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

Dimensão (n):

30

Número de Agentes:

50

Número de Iterações:

500

Número de Execuções:

30

Iniciar tarefa

Fonte: Elaborada pelo autor.

4.2.6 Detalhes da Tarefa

Na página de detalhes da tarefa, pode-se conferir a descrição dessa tarefa, área que conta com o identificador e o nome. Além disso, pode-se também acompanhar a execução em tempo real por meio da barra de progresso. Essas duas áreas podem ser visualizadas na [Figura 30](#). É importante frisar que, caso não haja núcleos de processamento disponíveis, a barra de progresso fica em um estado de espera, diferentemente do estado de progresso ilustrado. Ainda na mesma página, pode-se conferir uma área com os parâmetros de execução configurados bem como os limites empregados para o domínio da função de teste selecionada.

Essa área fica fechada por padrão, mas pode ser aberta por meio de um clique. Ademais, há, no fim da página, uma área para uma breve apresentação dos resultados ao término da tarefa. Essas duas últimas áreas podem ser observadas por meio da [Figura 31](#).

Figura 30 – Parte 1 dos Detalhes da Tarefa.

Descrição

Identificador	018df5b3-1c90-43b5-86ac-1c2795ffa094
Tipo	Otimização

Progresso

A execução acontece no servidor, logo você pode continuar navegando ou desconectar-se.

361 de 500 iterações. Execução 3...

Cancelar

Fonte: Elaborada pelo autor.

Figura 31 – Parte 2 dos Detalhes da Tarefa.

Parâmetros de Execução

Otimizador	Algoritmo Genético (GA)
Função objetivo	Função Powell Sum
Espaço de busca	Dimensão: 30 Limite inferior: -1 para todas as variáveis Limite superior: 1 para todas as variáveis
Número de agentes	30
Número de iterações	500
Número de execuções	5

Resultados

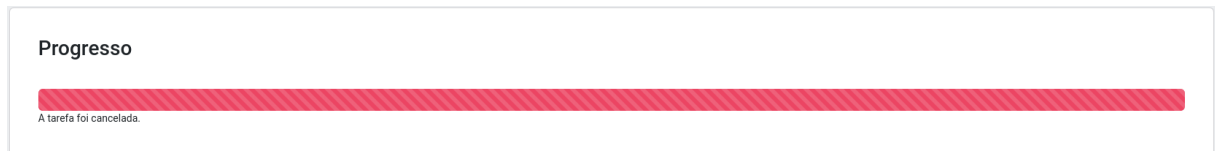
Se tudo ocorrer bem, os resultados aparecerão aqui após a execução.

Fonte: Elaborada pelo autor.

Como se pode notar nas imagens apresentadas, há um botão que permite o cancelamento da execução da tarefa. Caso esse botão seja pressionado, a tarefa é revogada e a área de

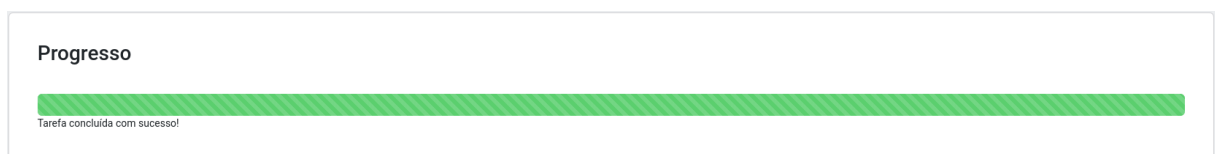
progresso assume o estado exibido pela [Figura 32](#). No entanto, se o usuário deixar a tarefa executar até o final, a barra de progresso muda para o estado apresentado na [Figura 33](#). Outrossim, quando a tarefa é concluída com sucesso, a área de resultados passa a exibir a melhor solução encontrada e o respectivo melhor valor da função conforme a [Figura 34](#).

Figura 32 – Barra de Progresso da Tarefa Cancelada.



Fonte: Elaborada pelo autor.

Figura 33 – Barra de Progresso da Tarefa Concluída.



Fonte: Elaborada pelo autor.

Figura 34 – Principais Resultados da Tarefa.

Resultados	
Melhor solução	[0,006; -0,056; 0,007; 0,131; 0,026; 0,185; 0,204; -0,208; -0,196; 0,037; 0,004; -0,294; -0,100; 0,436; -0,294; -0,149; 0,502; -0,010; -0,349; -0,382; -0,368; -0,054; 0,280; 0,025; -0,204; -0,033; 0,084; 0,161; -0,036; -0,144]
Melhor valor da função	2,749e-4

Ver mais

Fonte: Elaborada pelo autor.

4.2.7 Resultados

A penúltima página a ser discutida é a de resultados. Essa página contém informações diferentes de acordo com o número de execuções da tarefa, uma vez que, quando se executa uma tarefa mais de uma vez, pode-se produzir dados estatísticos. Em um primeiro momento, acompanha-se na [subseção 4.2.7.1](#), os resultados de uma única execução. Logo na sequência, na [subseção 4.2.7.2](#), apresenta-se a página de resultados para múltiplas execuções.

4.2.7.1 Execução Única

Na [Figura 35](#), apresenta-se a parte superior da página de resultados de uma única execução. Nessa parte, há uma área que contém a melhor solução encontrada, o respectivo melhor valor da função, a solução ótima e o respectivo valor da função. É importante frisar que a solução e o valor ótimo da função foram castrados no banco para que o usuário pudesse ter uma noção de quão próximo o otimizador chegou. Por sua vez, a [Figura 36](#) apresenta o gráfico de convergência da tarefa executada. Esse gráfico exibe os valores de função obtidos pelos melhores agentes do otimizador no decorrer das iterações.

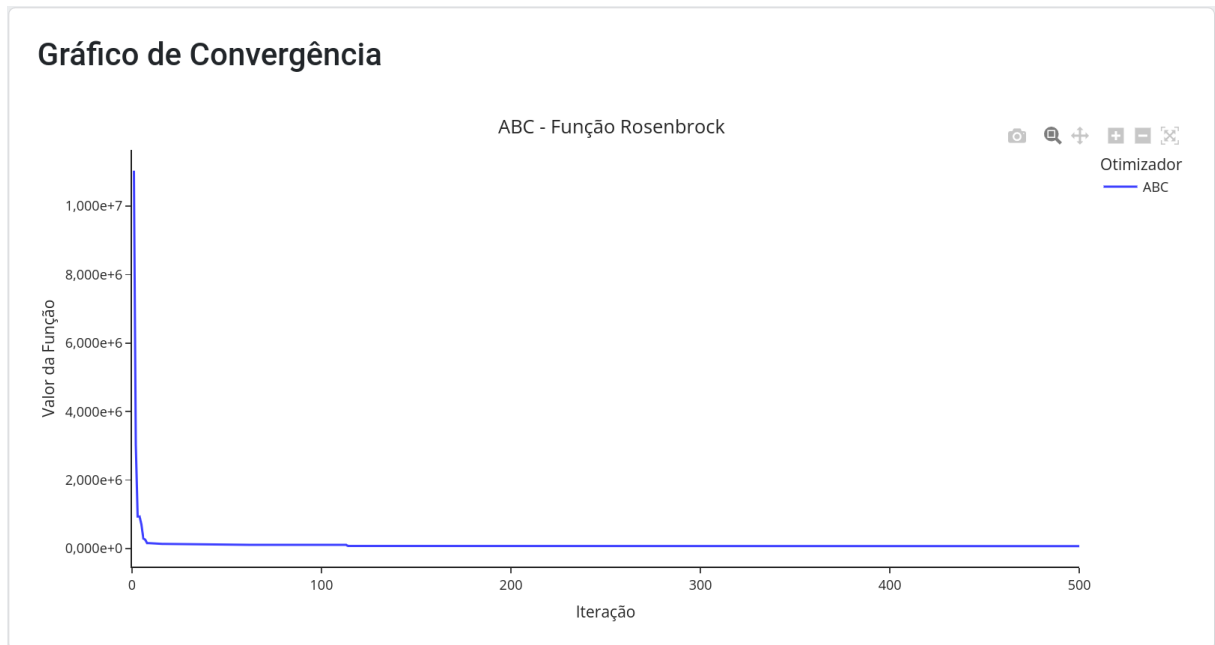
Ainda em relação ao gráfico, é possível observar que o Plotly, módulo empregado, adiciona um menu no canto superior direito, permitindo que o usuário salve em seu computador a imagem do gráfico, aumente ou diminua uma seção do gráfico etc. Embora não seja visível na imagem, esse módulo de geração de gráficos permite que o usuário acompanhe cada valor da curva conforme ele move o ponteiro do mouse através da curva.

Figura 35 – Resultados da Otimização.

Otimização	
Solução	
Melhor solução	[-1,443; -0,697; -2,258; 3,464; -0,303; 2,012; 2,052; -2,846; 1,031; -0,535; -2,747; 2,690; -3,158; 1,070; 3,343; -1,033; -2,059; -2,517; 3,152; 5,521]
Função	
Melhor valor	7,394e+04
Ótimo Global	
Solução ótima	[1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000; 1,000]
Valor ótimo da função	0,000e+00

Fonte: Elaborada pelo autor.

Figura 36 – Gráfico de Convergência.



Fonte: Elaborada pelo autor.

4.2.7.2 Múltiplas Execuções

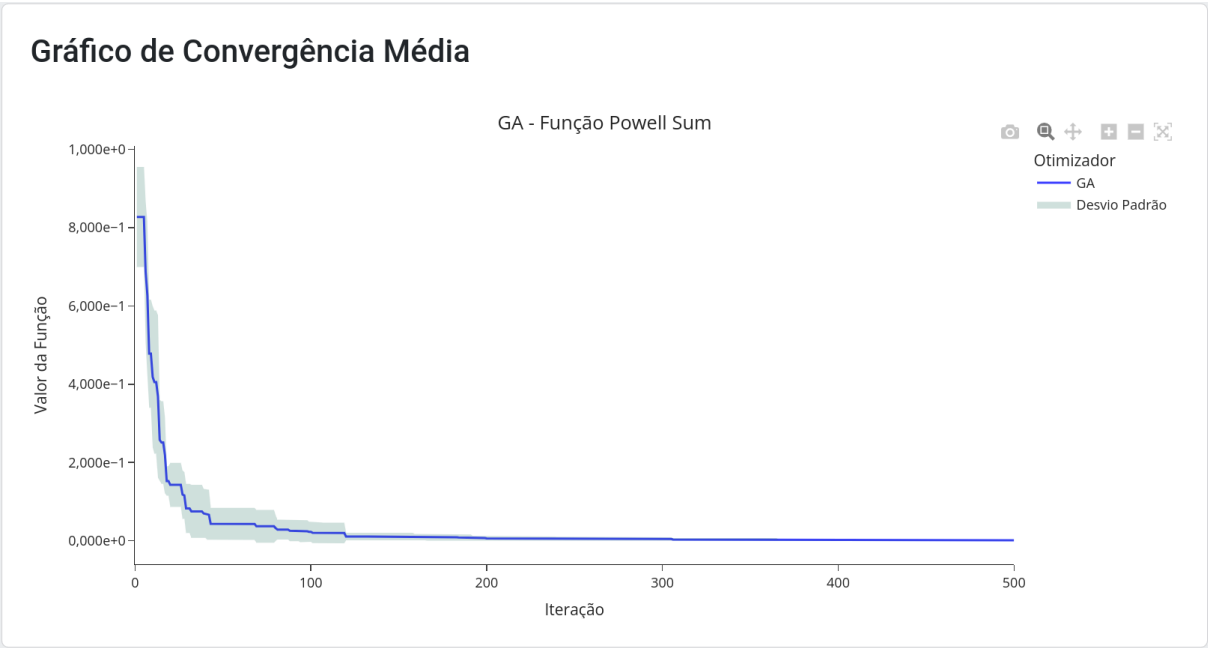
A [Figura 37](#) apresenta a porção superior dos resultados para casos de múltiplas execuções. Nessa parte, há uma área que contém a melhor solução encontrada, o respectivo melhor valor da função, o pior valor da função, o valor médio da função, o respectivo desvio padrão e o ótimo global. Em seguida, a [Figura 38](#) apresenta o gráfico de convergência média. Esse gráfico contém a curva, que foi obtida com a média dos melhores valores de função de cada iteração em cada execução, e a região de desvio padrão, que está sombreada ao redor da curva.

Figura 37 – Resultados da Otimização Com Estatísticas.

Otimização	
Solução	
Melhor solução	[0,006; -0,056; 0,007; 0,131; 0,026; 0,185; 0,204; -0,208; -0,196; 0,037; 0,004; -0,294; -0,100; 0,436; -0,294; -0,149; 0,502; -0,010; -0,349; -0,382; -0,368; -0,054; 0,280; 0,025; -0,204; -0,033; 0,084; 0,161; -0,036; -0,144]
Função	
Melhor valor	2,749e-04
Pior valor	3,626e-03
Valor médio	1,220e-03
Desvio padrão	1,385e-03
Ótimo Global	
Solução ótima	[0,000; 0,000]
Valor ótimo da função	0,000e+00

Fonte: Elaborada pelo autor.

Figura 38 – Gráfico de Convergência Média.



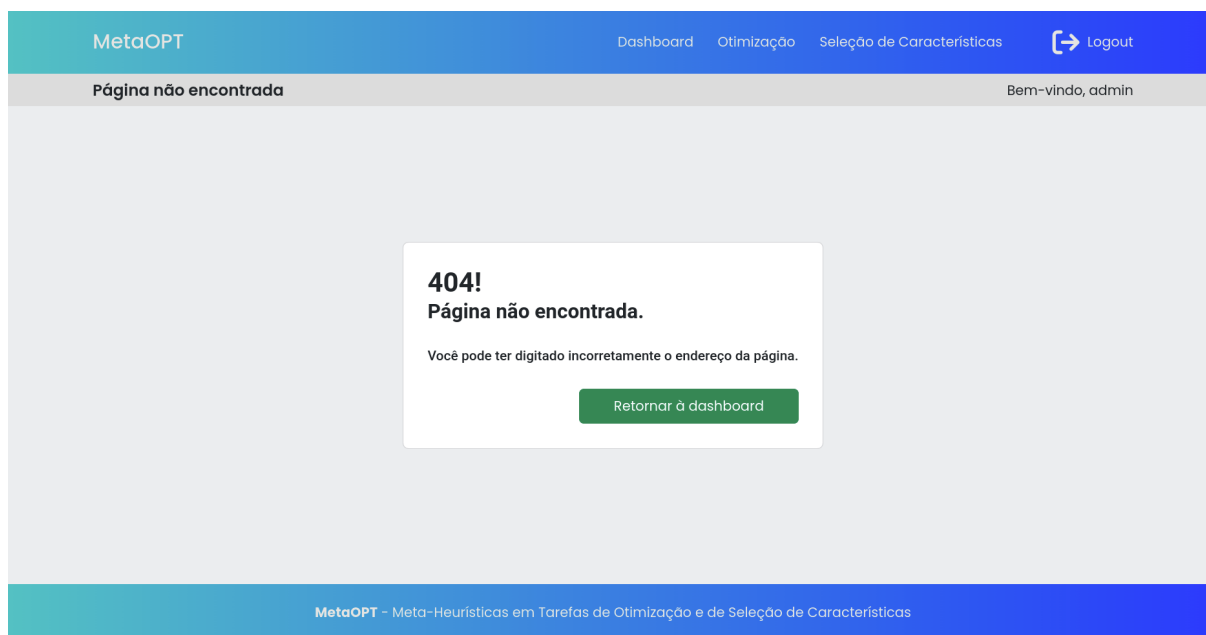
Fonte: Elaborada pelo autor.

4.2.8 *Página de Erro*

Por último, é interessante apontar que a aplicação realiza o tratamento dos possíveis erros que podem ocorrer durante a navegação dos usuários. Dentre esses erros, pode-se citar

dois bem famosos. O primeiro é o erro 404 e ocorre quando o caminho digitado pelo usuário não existe no domínio do aplicativo ou pertence a outro usuário, como no caso de uma página de detalhes de tarefa. Nesses casos, gera-se uma mensagem de página não encontrada como na Figura 39. Já o segundo é o erro 500, que representa um erro interno do servidor. É claro que esse último erro não deve ocorrer, porém é essencial remediar tais situações.

Figura 39 – Página de Erro 404



Fonte: Elaborada pelo autor.

5 Conclusão

Devido à crescente necessidade de gerenciar bem recursos como tempo e dinheiro na sociedade contemporânea, problemas de otimização têm sido cada vez mais estudados em áreas como a pesquisa operacional e a ciência da computação. No entanto, como exposto no trabalho, métodos tradicionais de otimização não lidam bem com os problemas complexos encontrados no mundo real. Sendo assim, otimizadores meta-heurísticos, capazes de encontrar soluções satisfatórias para inúmeros problemas, surgem como uma interessante alternativa.

No entanto, há uma abundância de técnicas bio-inspiradas promissoras que precisam ser estudadas para se conhecer o desempenho em certos tipos de problema. Dentre as ferramentas de meta-heurística, não há aplicações web que permitam o acesso de qualquer dispositivo sem a instalação de módulos. Dessa carência, nasce o projeto MetaOPT no laboratório de pesquisa Recogna em Bauru. Por sua vez, o presente trabalho assume a implementação do aplicativo de otimização desse projeto, permitindo que principiantes se familiarizem com os principais otimizadores meta-heurísticos através das respectivas convergências em alguns problemas artificiais de teste que são, frequentemente, empregados em análises de performance.

Por meio da análise de estudos realizados por autores da área de otimização via meta-heurísticas, selecionou-se seis técnicas populares e dez funções para os testes de performance. Com esses dados, criou-se uma aplicação web que viabiliza a execução assíncrona de tarefas de otimização, permitindo o acompanhamento do progresso em tempo real, a persistência dos dados de execução na base de dados do Django, a geração de gráficos de convergência e a apresentação de dados estatísticos, no caso de múltiplas execuções de uma tarefa.

5.1 Trabalhos Futuros

Tendo em mente que o aplicativo faz parte do projeto MetaOPT, pretende-se realizar a hospedagem o sistema e divulgar para que os integrantes do laboratório, bem como demais membros da comunidade acadêmica, possam executar tarefas de otimização. Com esses utilizadores, tem-se a possibilidade de buscar opiniões a respeito de possíveis incrementos. No entanto, é possível mencionar interessantes acréscimos para rebuscar a aplicação.

Em primeiro lugar, tem-se a criação de uma página para comparação das tarefas já executadas. Com essa página, seria possível selecionar tarefas com parâmetros semelhantes, mas que utilizam diferentes técnicas. Nessa página, pode-se criar uma tabela contrapondo os resultados e produzir um gráfico de convergência com as respectivas curvas.

Ademais, em relação ao formulário para criação de novas tarefas de otimização, pode-se permitir que o usuário atribua uma descrição para cada tarefa, facilitando ainda mais a posterior

localização dessa tarefa. Outrossim, ainda nesse formulário, seria pertinente disponibilizar mais opções tanto de otimizadores quanto de funções de teste. Em relação às funções, também seria interessante permitir que o usuário definisse seus próprios problemas.

Por fim, há duas adições atrativas relacionadas à execução. A primeira é uma métrica para contabilizar o tempo de execução das tarefas. Já a segunda é a possibilidade de pausar e retomar a execução das tarefas, mecanismo relevante para a obtenção de resultados preliminares e já suportado pela biblioteca Opytimizer, que viabiliza a execução das meta-heurísticas.

Referências

- BOOTSTRAP. *Build fast, responsive sites with Bootstrap*. Bootstrap, 2023. Página inicial do Bootstrap. Disponível em: <<https://getbootstrap.com/>>. Acesso em: 02 jan. 2023.
- CASTELEYN, S.; DANIEL, F.; DOLOG, P.; MATERA, M. *Engineering Web Applications*. 1. ed. [S.l.]: Springer Berlin Heidelberg, 2009. ISSN 2197-9723.
- CELERY. *Celery - Distributed Task Queue*. Celery, 2023. Documentação do Celery 5.2.7. Disponível em: <<https://docs.celeryq.dev/en/stable/>>. Acesso em: 02 jan. 2023.
- DOKEROGLU, T.; SEVINC, E.; KUCUKYILMAZ, T.; COSAR, A. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, Elsevier Ltd., v. 137, p. 106040, 2019. ISSN 0360-8352. Disponível em: <<https://www.sciencedirect.com/science/article/abs/pii/S0360835219304991>>. Acesso em: 02 jan. 2023.
- HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. 2. ed. Cambridge, USA: MIT Press, 1992. ISBN 0262082136.
- HUSSAIN, K.; SALLEH, M. N. M.; CHENG, S.; NASEEM, R. Common Benchmark Functions for Metaheuristic Evaluation: A Review. *JOIV : International Journal on Informatics Visualization*, Politeknik Negeri Padang, v. 1, n. 4-2, p. 218–223, nov 2017. ISSN 2549-9904. Disponível em: <<http://joiv.org/index.php/joiv/article/view/65>>. Acesso em: 02 jan. 2023.
- JAMIL, M.; YANG, X. S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, Inderscience Publishers, v. 4, n. 2, p. 150–194, 2013. ISSN 20403615. Disponível em: <<https://arxiv.org/abs/1308.4008>>. Acesso em: 02 jan. 2023.
- KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, v. 39, n. 3, p. 459–471, oct 2007. ISSN 1573-2916. Disponível em: <<http://link.springer.com/10.1007/s10898-007-9149-x>>. Acesso em: 02 jan. 2023.
- KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Perth, Australia: IEEE, 1995. v. 4, p. 1942–1948. ISBN 0-7803-2768-3. Disponível em: <<https://ieeexplore.ieee.org/document/488968>>. Acesso em: 02 jan. 2023.
- KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, v. 220, n. 4598, p. 671–680, 1983. Disponível em: <<https://www.science.org/doi/abs/10.1126/science.220.4598.671>>. Acesso em: 02 jan. 2023.
- PAREJO, J. A.; RUIZ-CORTÉS, A.; LOZANO, S.; FERNANDEZ, P. Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Computing*, Springer, v. 16, n. 3, p. 527–561, mar 2012. ISSN 14327643. Disponível em: <<https://link.springer.com/article/10.1007/s00500-011-0754-8>>. Acesso em: 02 jan. 2023.

PLOTLY. *Plotly Python Graphing Library*. Plotly, 2023. Documentação do Plotly. Disponível em: <<https://plotly.com/python/>>. Acesso em: 02 jan. 2023.

RAMÍREZ, A.; BARBUDO, R.; ROMERO, J. R. An experimental comparison of metaheuristic frameworks for multi-objective optimization. *Expert Systems*, John Wiley & Sons, Ltd, p. e12672, 2021. ISSN 1468-0394. Disponível em: <<https://onlinelibrary.wiley.com/doi/epdf/10.1111/exsy.12672>>. Acesso em: 02 jan. 2023.

RAO, S. S. *Engineering Optimization: Theory and Practice*. 4. ed. [S.l.]: John Wiley and Sons, 2009. ISBN 9780470183526.

REDIS. *Redis Message Broker*. Redis, 2023. Página inicial do Redis. Disponível em: <<https://redis.com/solutions/use-cases/messaging/>>. Acesso em: 02 jan. 2023.

ROSA, G. H. de; RODRIGUES, D.; PAPA, J. P. Opytimizer: A Nature-Inspired Python Optimizer. arXiv, 2019. Disponível em: <<https://arxiv.org/abs/1912.13002v2>>. Acesso em: 02 jan. 2023.

YANG, X. S. Firefly Algorithms for Multimodal Optimization. In: WATANABE, O.; ZEUGMANN, T. (Ed.). *Stochastic Algorithms: Foundations and Applications*. Springer Berlin Heidelberg, 2009. v. 5792, p. 169–178. ISBN 978-3-642-04944-6. Disponível em: <http://link.springer.com/10.1007/978-3-642-04944-6_14>. Acesso em: 02 jan. 2023.

YANG, X. S. *Nature-Inspired Optimization Algorithms*. [S.l.]: Elsevier Inc., 2014. ISBN 9780124167438.

YANG, X. S.; DEB, S. Cuckoo search via lévy flights. In: *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*. Coimbatore, India: IEEE, 2009. p. 210–214. ISBN 978-1-4244-5053-4. Disponível em: <<http://ieeexplore.ieee.org/document/5393690/>>. Acesso em: 02 jan. 2023.