

fabio-linhares /
occ-2024-2[Code](#)[Issues](#)[Pull requests](#)[Actions](#)[Projects](#)[Wiki](#)[Security](#)[otimizacao](#) [occ-2024-2 / b1 / README.md](#) 

fabio-linhares fix: corrigir links de imagens no README.md

e361303 · 1 hour ago



300 lines (226 lo...)

Meta-heurística para o Problema de Bin Packing

Este projeto implementa uma meta-heurística híbrida para resolver o problema de Bin Packing de forma eficiente. A aplicação inclui uma interface web interativa para visualização e experimentação com diferentes instâncias do problema.

O Problema de Bin Packing

O problema de Bin Packing consiste em empacotar um conjunto de itens com dimensões variadas em recipientes (bins) de capacidade fixa, minimizando o número total de recipientes utilizados. Este é um problema NP-difícil clássico com aplicações em diversas áreas como logística, manufatura e alocação de recursos.

Nossa Abordagem

Implementamos uma meta-heurística híbrida que combina:

- Busca Local (Best/First Improvement)
- Simulated Annealing
- Iterated Local Search com perturbações adaptativas
- Estratégias de reinício

Características

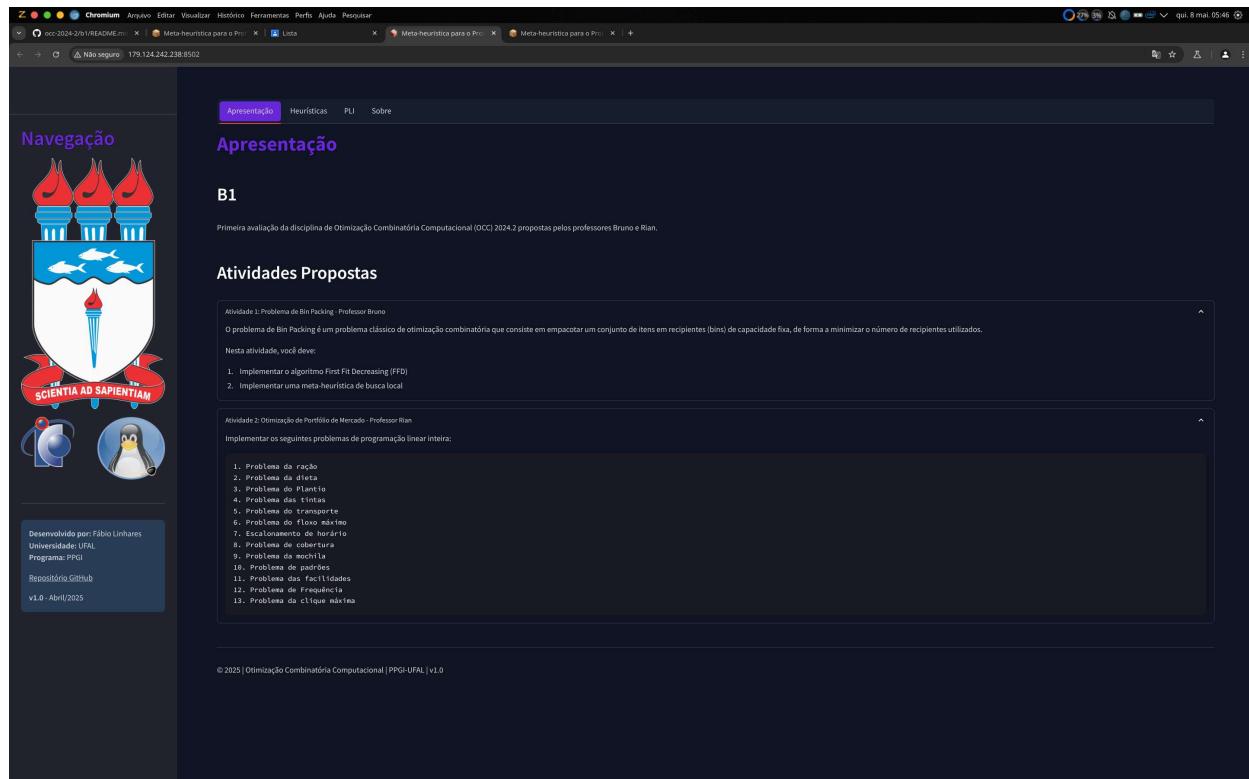
- Inicialização com First-Fit Decreasing

- Múltiplos tipos de perturbação para escapar de ótimos locais
- Visualização colorida dos itens em cada bin
- Métricas detalhadas sobre a qualidade da solução
- Interface interativa para carregamento de instâncias e configuração do algoritmo

Screenshots e Exemplos

Aqui estão alguns exemplos de visualizações e resultados obtidos com nossa implementação:

Interface Principal



Apresentação

Navegação



SCIENTIA AD SAPIENTIAM

Desenvolvido por: Fábio Linhares
Universidade: UFAL
Programa: PPGI
Repositório GitHub
v1.0 - Abril/2025

Heurísticas para Bin Packing

Apresentação Heurísticas PLI Sobre

Apresentação Solução

Meta-heurísticas para o Problema de Bin Packing

Fundamentação Teórica

O problema de Bin Packing (BPP) é um problema de otimização combinatória classificado como NP-difícil no sentido forte. Formalmente, o problema pode ser definido como:

Dado um conjunto de n itens, cada um com um tamanho $s_i \in (0, 1]$, e m bins (recipientes) idênticos de capacidade 1, o objetivo é empacotar todos os itens utilizando o menor número possível de bins, garantindo que a soma dos tamanhos dos itens em cada bin não excede a capacidade.

Formulação Matemática

O BPP pode ser formulado como um problema de Programação Linear Inteira:

$$\min \sum_{j=1}^m y_j$$

Sujeito a:

$$\sum_{j=1}^m x_{ij} = 1, \quad \forall i \in \{1, 2, \dots, n\}$$

$$\sum_{j=1}^m s_i x_{ij} \leq y_j, \quad \forall j \in \{1, 2, \dots, m\}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in \{1, 2, \dots, n\}, \forall j \in \{1, 2, \dots, m\}$$

$$y_j \in \{0, 1\}, \quad \forall j \in \{1, 2, \dots, m\}$$

Onde:

- $y_j = 1$ se o bin j for utilizado, 0 caso contrário
- $x_{ij} = 1$ se o item i for alocado ao bin j , 0 caso contrário
- s_i é o tamanho do item i
- m é um limite superior para o número de bins (geralmente $m = n$)

Estado da Arte em Heurísticas para BPP

Devido à natureza NP-difícil do problema, várias heurísticas foram desenvolvidas para encontrar soluções aproximadas em tempo computacional razoável.

Heurísticas Construtivas Clássicas

Meta-heurísticas Avançadas

Navegação



SCIENTIA AD SAPIENTIAM

Desenvolvido por: Fábio Linhares
Universidade: UFAL
Programa: PPGI
Repositório GitHub
v1.0 - Abril/2025

Estado da Arte em Heurísticas para BPP

Devido à natureza NP-difícil do problema, várias heurísticas foram desenvolvidas para encontrar soluções aproximadas em tempo computacional razoável.

Heurísticas Construtivas Clássicas

- First Fit (FF)
 - Coloca cada item no primeiro bin onde cabe
 - Complexidade: $O(n \log n)$
- Best Fit (BF)
 - Coloca cada item no bin mais cheio onde ele ainda cabe
 - Complexidade: $O(n \log n)$
- First Fit Decreasing (FFD)
 - Ordena os itens em ordem decrescente de tamanho
 - Aplica First Fit à lista ordenada
 - Garante teórica: $FFD(I) \leq \frac{11}{10} OPT(I) + 1$
- Best Fit Decreasing (BFD)
 - Ordena os itens em ordem decrescente de tamanho
 - Aplica Best Fit à lista ordenada
 - Desempenho similar ao FFD na prática

Meta-heurísticas Avançadas

- Busca Local
 - Explora o espaço de soluções através de pequenas modificações
 - Estruturas de vizinhança: movimentos de itens entre bins
- Simulated Annealing (SA)
 - Inspirado no processo físico de resfriamento de metais
 - Aceta soluções piores com probabilidade controlada para escapar de ótimos locais
 - Probabilidade de aceitação: $P = e^{-\Delta/T}$
- Iterated Local Search (ILS)
 - Aplica perturbações quando a busca local estagnar
 - Permite explorar diferentes regiões do espaço de busca
- Algoritmos Genéticos e Evolutivos
 - Utilizam operadores inspirados na evolução natural
 - Crossover, mutação e seleção para evolução da população

Nossa Abordagem: Meta-heurística Híbrida

Implementamos uma meta-heurística híbrida que combina elementos de diferentes abordagens para obter um algoritmo mais robusto e eficiente:

Componentes da Meta-heurística

- Inicialização
 - Solução inicial gerada com First Fit Decreasing (FFD)
- Busca Local
 - Exploração sistemática da vizinhança através de movimentos de itens entre bins
 - Dois tipos de estratégia: Best Improvement e First Improvement
- Mecanismos para Escapar de Ótimos Locais
 - Simulated Annealing: Aceta soluções piores com probabilidade $P = e^{-\Delta/T}$
 - Perturbações Inteligentes: Quatro tipos de perturbações com intensidade adaptativa
 - Estratégia de Reinição: Quando não há progresso após várias iterações

Tipos de Perturbação Implementados

Esquema de Resfriamento (SA)

Navegação

Tipos de Perturbação Implementados

1. Move
 - Move um item aleatório de um bin para outro
 - Perturbação de baixa intensidade
2. Swap
 - Troca itens entre dois bins diferentes
 - Mantém o número de bins constante
3. Redistribute
 - Redistribui todos os itens de um bin para outros bins
 - Potencial para reduzir o número total de bins
4. Merge-Split
 - Combina dois bins e redistribui os itens ottimamente
 - Perturbação de alta intensidade que pode levar a reorganizações significativas

Exemplo de empacotamento de itens em bins

Comparação Teórica de Desempenho

Para instâncias com n itens, as complexidades teóricas são:

Algoritmo	Complexidade	Razão de Aproximação
First Fit	$O(n \log n)$	$FF(I) \leq 1.7 \times OPT(I) + 0.7$
Best Fit	$O(n \log n)$	$BF(I) \leq 1.7 \times OPT(I) + 0.7$
First Fit Decreasing	$O(n \log n)$	$FFD(I) \leq \frac{11}{3} \times OPT(I) + 1$
Nossa Meta-heurística	Depende do tempo limite	Não há garantia teórica, mas empiricamente obtém soluções próximas do ótimo

A abordagem híbrida implementada consegue escapar de ótimos locais e explorar eficientemente o espaço de busca, resultando em soluções de alta qualidade para o problema de Bin Packing.

Referências

1. Martello, S., & Toth, P. (1990). Knapsack problems: algorithms and computer implementations. John Wiley & Sons.
2. Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. *Handbook of combinatorial optimization*, 455-531.
3. Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. *Handbook of metaheuristics*, 320-353.
4. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680.

Otimização de empacotamento de itens utilizando busca local

Navegação

Referências

1. Martello, S., & Toth, P. (1990). Knapsack problems: algorithms and computer implementations. John Wiley & Sons.
2. Coffman Jr, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin packing approximation algorithms: survey and classification. *Handbook of combinatorial optimization*, 455-531.
3. Lourenço, H. R., Martin, O. C., & Stützle, T. (2003). Iterated local search. *Handbook of metaheuristics*, 320-353.
4. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680.

Otimização de empacotamento de itens utilizando busca local

Esta aplicação resolve o problema do Bin Packing utilizando uma meta-heurística baseada em busca local. O objetivo é encontrar a melhor distribuição dos itens em recipientes, minimizando o número total de recipientes utilizados. A partir de uma solução inicial gerada pelo algoritmo First Fit Decreasing, a busca local tenta melhorar a solução através de movimentos de troca e realocação de itens.

Problema de Bin Packing

O problema de Bin Packing é um problema clássico de otimização combinatória que consiste em empacotar um conjunto de itens em recipientes (bins) de capacidade fixa, de forma a minimizar o número de recipientes utilizados.

Aplicações

Este problema possui diversas aplicações práticas, como:

- Alocação de contêineres em navios
- Armazenamento de dados em discos
- Corte de materiais para minimizar desperdício
- Balanceamento de carga em servidores
- Empacotamento de produtos para transporte

Formulação Matemática

O Bin Packing pode ser formulado como um problema de programação linear inteira:

Variáveis de decisão:

- $y_{j|i}$: 1 se o bin j é utilizado, 0 caso contrário
- $x_{i|j}$: 1 se o item i é aloocado no bin j , 0 caso contrário

Função objetivo:

$$\min \sum_{j=1}^n y_j$$

Minimizar o número total de bins utilizados

Formulação Matemática

O Bin Packing pode ser formulado como um problema de programação linear inteira:

Variáveis de decisão:

- y_{jS} : 1 se o bin S é utilizado, 0 caso contrário
- $x_{i(j)}$: 1 se o item i é alocado no bin S , 0 caso contrário

Função objetivo:

$$\min \sum_{j \in J} y_j$$

Minimizar o número total de bins utilizados

Restrições:

- Cada item deve ser atribuído a exatamente um bin:
$$\sum_{j \in J} x_{i(j)} = 1, \quad \forall i \in I$$

- A capacidade dos bins não pode ser excedida:
$$\sum_{i \in I} w_i x_{i(j)} \leq C, \quad \forall j \in J$$

- Restrições de integralidade:
$$x_{i(j)} \in \{0,1\}, \quad \forall i \in I, \forall j \in J$$

Onde:

- I : conjunto de itens
- S : conjunto de bins (no pior caso, um bin para cada item)
- w_i : tamanho (peso) do item i
- C : capacidade de cada bin

Heurísticas Implementadas

First Fit Decreasing (FFD)

O algoritmo First Fit Decreasing é uma heurística construtiva que funciona da seguinte forma:

- Ordena os itens em ordem decrescente de tamanho
- Para cada item, tenta coloca-lo no primeiro bin onde ele couber
- Se não couber em nenhum bin existente, cria um novo bin

Pseudocódigo:

```
Função FirstFitDecreasing(itens):
```



Desenvolvido por: Fábio Linhares
Universidade: UFLA
Programa: PPGI
Repositório GitHub
v1.0 - Abril/2025

Heurísticas Implementadas

First Fit Decreasing (FFD)

O algoritmo First Fit Decreasing é uma heurística construtiva que funciona da seguinte forma:

- Ordena os itens em ordem decrescente de tamanho
- Para cada item, tenta coloca-lo no primeiro bin onde ele couber
- Se não couber em nenhum bin existente, cria um novo bin

Pseudocódigo:

```
Função FirstFitDecreasing(itens):
```

```
    Ordena_itens_por_tamanho (decrescente)
```

```
    bins = []
```

```
    Para cada item em itens:
```

```
        colocado = falso
```

```
        Para cada bin em bins:
```

```
            Se bin.capacidade_restante >= item.tamanho:
```

```
                Adiciona item ao bin
```

```
                colocado = verdadeiro
```

```
                Break
```

```
            Se não colocado:
```

```
                Cria novo bin
```

```
                Adiciona item ao novo bin
```

```
                Adiciona novo bin à lista de bins
```

```
    Retorna bins
```

Busca Local

A meta-heurística de busca local implementada explora a vizinhança de uma solução inicial (gerada pelo FFD) através de dois tipos de movimentos:

- Movimento de Realocação: move um item de um bin para outro
- Movimento de Troca: troca itens entre dois bins

A função de avaliação considera tanto o número de bins utilizados (componente principal) quanto o balançoamento de carga dos bins (componente secundário).

Processo de busca:

- Começa com a solução inicial gerada pelo FFD
- Explora todos os movimentos possíveis da vizinhança
- Seleciona o melhor movimento (que reduz o valor da função de avaliação)
- Aplica o movimento, gerando uma nova solução
- Repete o processo até não encontrar mais movimentos de melhoria ou atingir o limite de tempo

Resultados Esperados

Navegação

Pseudocódigo:

```

Função FirstFitDecreasing(itens):
    Ordena itens por tamanho (decrescente)
    bins = []

    Para cada item em itens:
        colocado = false

        Para cada bin em bins:
            Se bin.capacidade_restante >= item.tamanho:
                Adiciona item ao bin
                colocado = verdadeiro
                Break

        Se não colocado:
            Cria novo bin
            Adiciona item ao novo bin
            Adiciona novo bin à lista de bins

    Retorna bins

```

Busca Local

A meta-heurística de busca local implementada explora a vizinhança de uma solução inicial (gerada pelo FFD) através de dois tipos de movimentos:

1. Movimento de Realocação: move um item de um bin para outro
2. Movimento de Troca: troca items entre dois bins

A função de avaliação considera tanto o número de bins utilizados (componente principal) quanto o balanceamento de carga dos bins (componente secundário).

Processo de busca:

1. Começa com a solução inicial gerada pelo FFD
2. Explora todos os movimentos possíveis da vizinhança
3. Seleciona o melhor movimento (que reduz o valor da função de avaliação)
4. Aplica o movimento, gerando uma nova solução
5. Repete o processo até não encontrar mais movimentos de melhoria ou atingir o limite de tempo

Resultados Esperados

A combinação do algoritmo FFD com a busca local permite encontrar soluções de boa qualidade para o problema de Bin Packing em tempo computacional razoável.

- O FFD fornece uma solução inicial rápida e de qualidade razoável
- A busca local refina essa solução, tentando reduzir o número de bins utilizados

Na aba "Bin Packing" é possível testar esses algoritmos em instâncias predefinidas ou geradas aleatoriamente.

© 2025 Fábio Linhares | [GitHub](#) | Meta-heurística para o Problema de Bin Packing | v1.0

Solução

Seleção da instância

Navegação

Heurísticas para Bin Packing

Meta-heurística de Solução Única para Bin Packing

Esta implementação resolve o problema de Bin Packing utilizando uma meta-heurística baseada em busca local. Você pode carregar um arquivo com instâncias do problema e definir um tempo limite para a execução.

Formato do arquivo de entrada:

- Cada linha contém um número entre 0 e 1, representando o tamanho de um item.
- Exemplo: 0.42, 0.25, 0.15, 0.31, 0.12, 0.5, 0.18

Carrregar arquivo de instância

Drag and drop file here
Limit 200MB per file • TXT, CSV

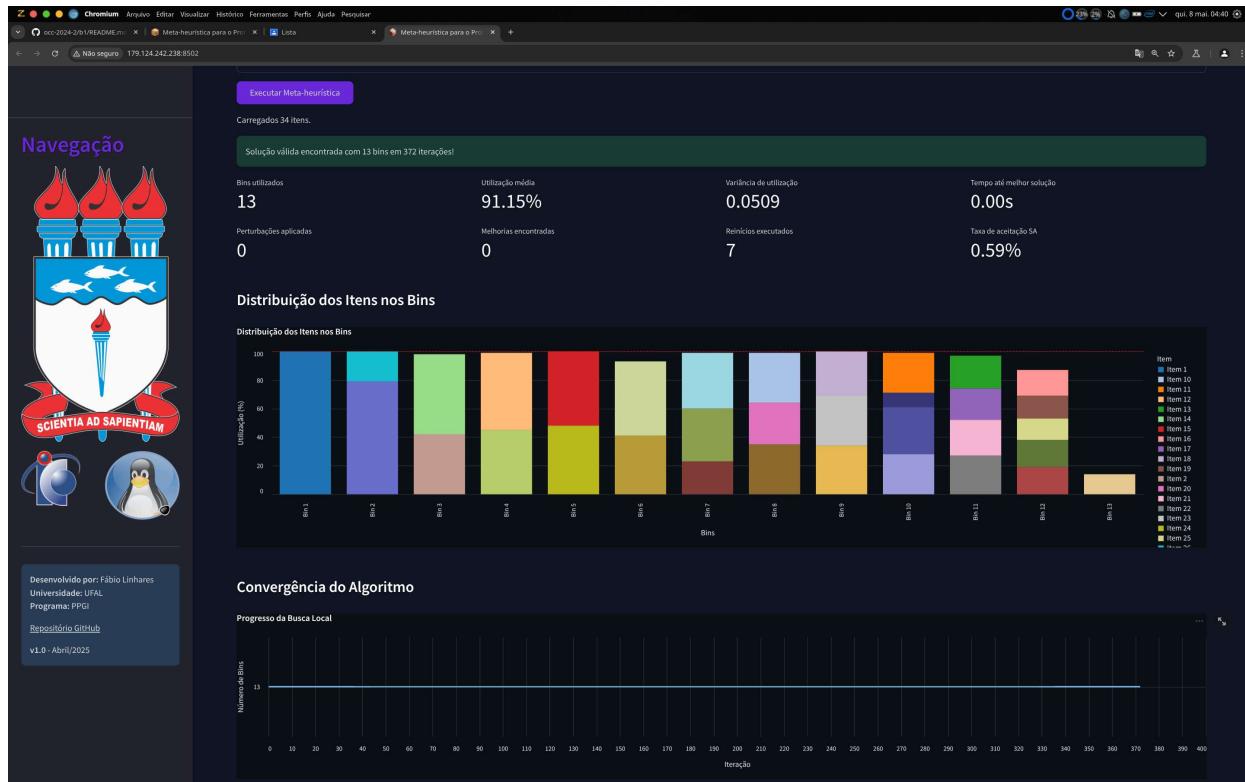
Tempo limite (segundos): Número máximo de bins:

Estratégia de busca local:
 Best Improvement
 First Improvement

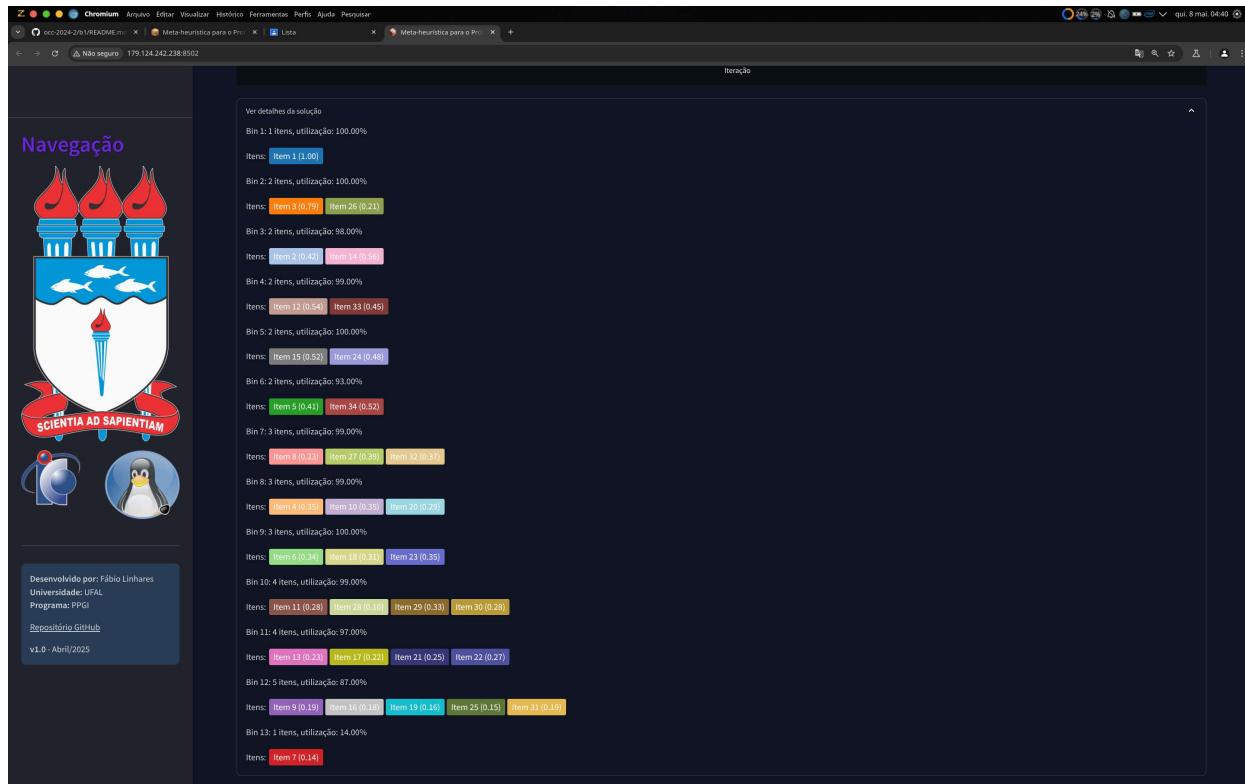
Opcionais:
 Usar Simulated Annealing
 Visualização detalhada

Desenvolvido por: Fábio Linhares
Universidade: UFAL
Programa: PPGI
Repositório GitHub
v1.0 - Abril/2025

Exibição dos bins



Detalhes dos bins



Programação Linear Inteira (ILP)

Além da meta-heurística, este projeto também implementa soluções exatas utilizando Programação Linear Inteira (ILP) para uma variedade de problemas clássicos de otimização. A implementação usa a biblioteca PuLP para modelar e resolver os problemas.

Como Utilizar

1. Acesse a aba "PLI" na interface da aplicação
2. Selecione o problema que deseja resolver
3. Carregue os arquivos de entrada correspondentes ao problema
4. Clique em "Resolver" para obter a solução ótima

A implementação utiliza o solver CBC através do PuLP, que é capaz de resolver problemas de médio porte com eficiência.

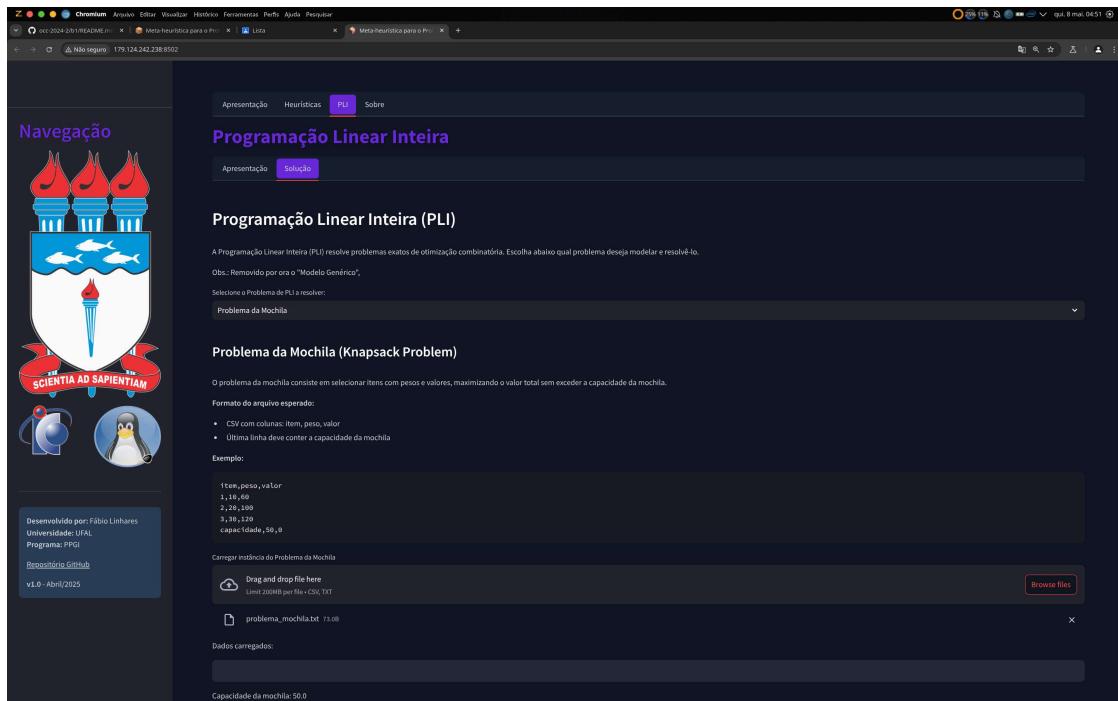
Problemas Implementados

Nossa implementação inclui os seguintes problemas de otimização:

1. Problema da Mochila (Knapsack Problem)

- **Descrição:** Selecionar itens com pesos e valores para maximizar o valor total sem exceder a capacidade da mochila.
- **Arquivo:** problema_mochila.txt - Contém itens com seus pesos e valores, e a capacidade da mochila.
- **Formato:** CSV com colunas: item, peso, valor (e uma linha para capacidade).

Screenshots:



2. Problema da Dieta

- **Descrição:** Selecionar alimentos minimizando o custo total enquanto atende requisitos nutricionais.
- **Arquivos:**
 - dieta_alimentos.txt - Alimentos com custos e valores nutricionais.
 - dieta_requisitos.txt - Requisitos mínimos e máximos de cada nutriente.
- **Formato:** CSVs com informações nutricionais e limites.
- **Screenshots:**

Programação Linear Inteira (PLI)

A Programação Linear Inteira (PLI) resolve problemas exatos de otimização combinatória. Escolha abaixo qual problema deseja modelar e resolvê-lo.

Obs.: Removido por ora o "Modelo Gênero".

Selecione o Problema de PLI a resolver:

Problema da Dieta

Problema da Dieta

O problema da dieta consiste em selecionar alimentos que satisfaçam requisitos nutricionais minimizando o custo total.

Formato dos arquivos esperados:

- Arquivo de Alimentos:
 - CSV com colunas: alimento, custo, e uma coluna para cada nutriente
 - Exemplo:

```
alimento,custo,proteina,lipideos,carbohidratos,calorias
Arroz,2,38,2,5,0,28,0,124
Feijao,5,50,7,0,8,15,14,0,77
Carne,25,80,26,0,15,0,0,8,288
```
- Arquivo de Requisitos Nutricionais:
 - CSV com colunas: nutriente, minimo, maximo
 - Exemplo:

```
nutriente,minimo,maximo
proteina,50,100
lipideos,30,50
carbohidratos,200,300
calorias,1800,2200
```

Carregar arquivo de alimentos

Drag and drop file here
Limit 20MB per file + CSV, TXT

Browse files

Carregar arquivo de requisitos nutricionais

Drag and drop file here
Limit 20MB per file + CSV, TXT

Browse files

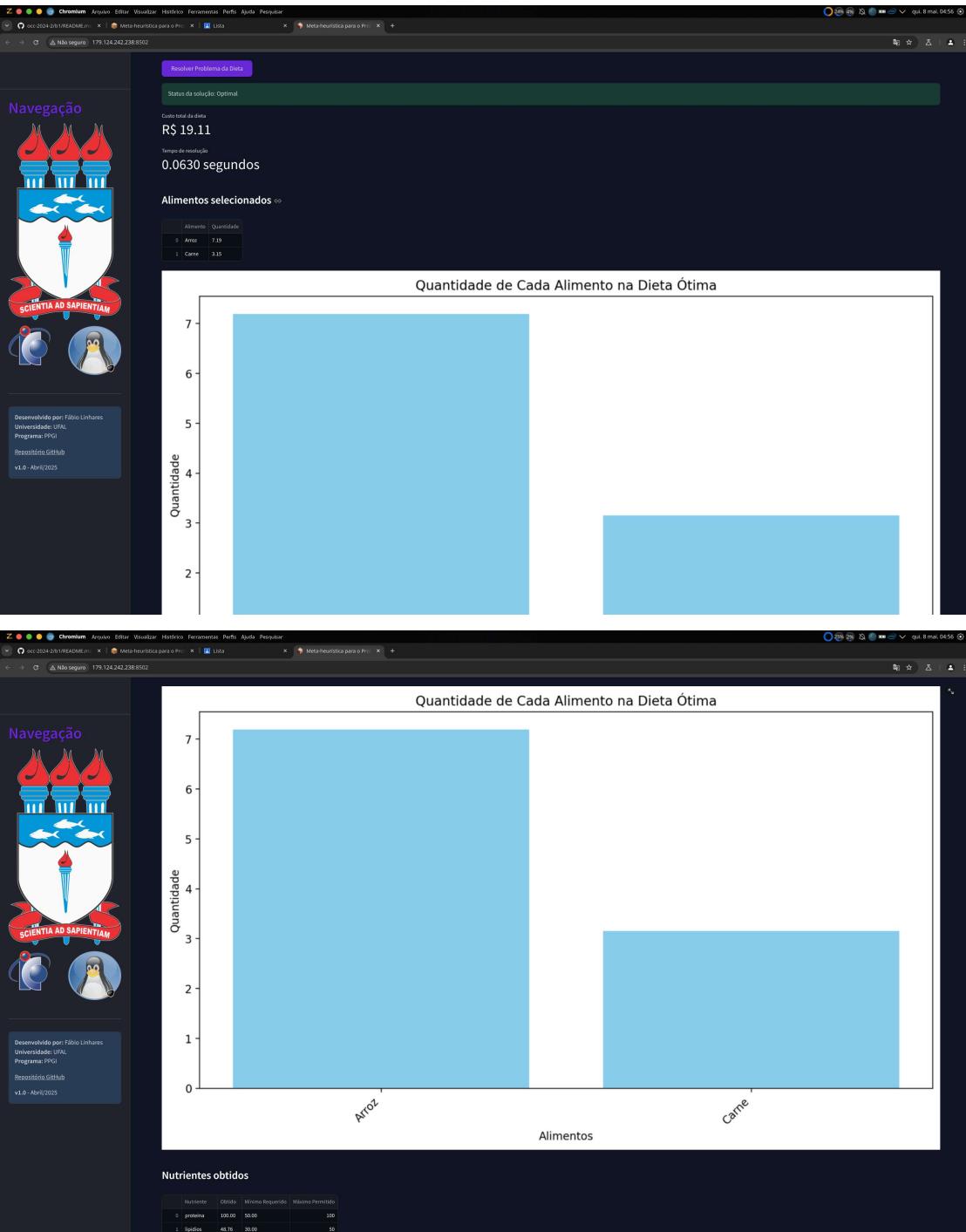
Dados de Alimentos

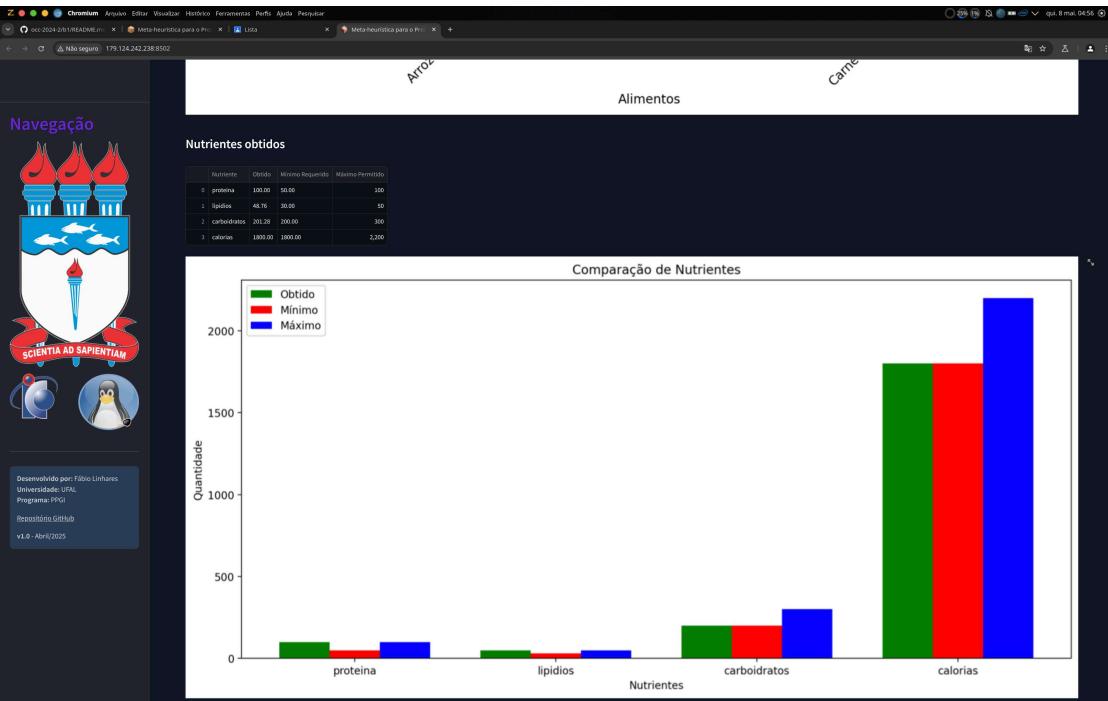
	alimento	custo	proteina	lipideos	carbohidratos	calorias
0	Arroz	2	2,5	0,2	28	124
1	Feijao	5	7	0,5	14	77
2	Carne	25	80	26,0	0,15,0	8,288
3	Leite	1	3,2	3,5	5	65

Requisitos Nutricionais

	nutriente	minimo	maximo
0	proteina	50	100
1	lipideos	30	50
2	carbohidratos	200	300
3	calorias	1,800	2,200

Resolver Problema da Dieta

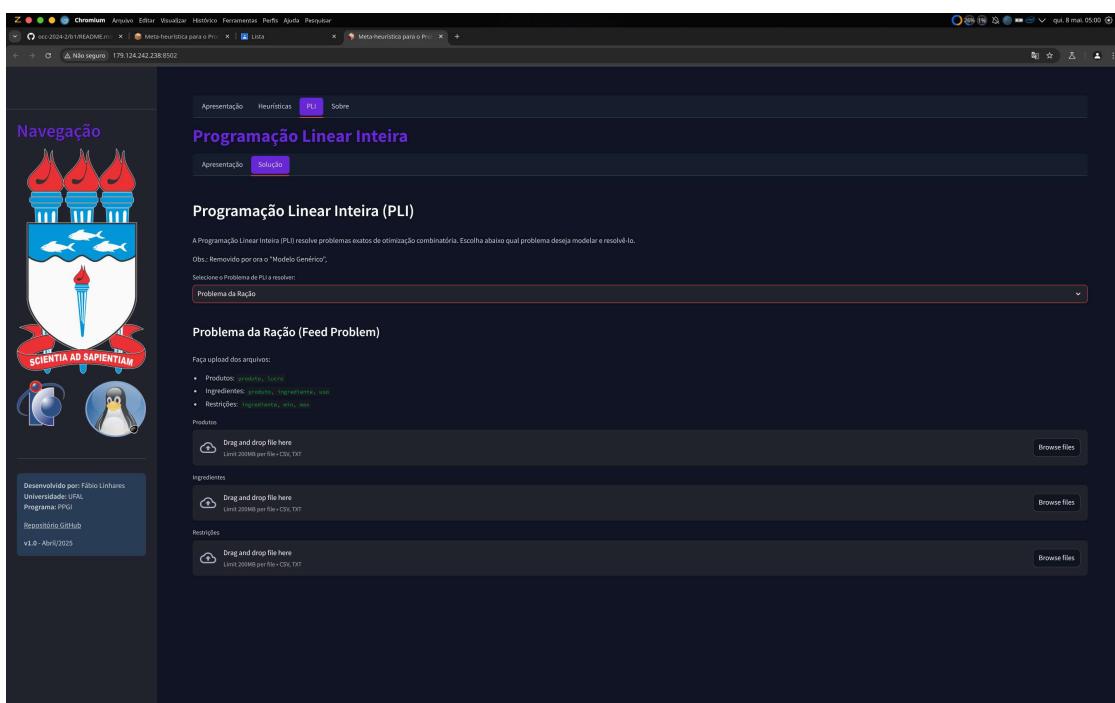




3. Problema da Ração

- **Descrição:** Determinar a produção ótima de diferentes tipos de ração para maximizar o lucro.
- **Arquivos:**
 - racao_produtos.txt - Produtos e seus lucros.
 - racao_ingredientes.txt - Relação entre produtos e ingredientes.
 - racao_restricoes.txt - Restrições de ingredientes.
- **Formato:** CSVs com relações entre produtos, ingredientes e restrições.

Screenshots:



4. Problema do Caixeiro Viajante (TSP)

- **Descrição:** Encontrar o caminho mais curto que visita todas as cidades exatamente uma vez.
- **Arquivo:** problema_caixeiro_viajante.txt - Matriz de distâncias entre cidades.
- **Formato:** CSV com matriz de distâncias (linhas e colunas representando cidades).

Screenshots:

The screenshot shows the "Programação Linear Inteira (PLI)" section. A shield logo with three red flames and the text "SCIENTIA AD SAPIENTIAM" is displayed. Below it are icons of a blue sphere and a penguin. The main content area is titled "Programação Linear Inteira (PLI)". It includes a sub-section "Problema do Caixeiro Viajante (TSP)". A CSV file example is shown:

```
cidades,A,B,C,D,E
A,0,15,15,20,0
B,15,0,12,15,16
C,15,12,0,6,14
D,20,15,6,0,10
E,0,15,14,10,0
```

Below this is a "Carregar matriz de distâncias (CSV)" section with a file input field containing "problema_caixeiro_viajante.csv". A "Browse files" button is also present. A "Matriz de distâncias:" table follows:

cidade	A	B	C	D	E
A	0	15	15	20	0
B	15	0	12	15	16
C	15	12	0	6	14
D	20	15	6	0	10
E	0	15	14	10	0

A "Resolver TSP" button is at the bottom.

The screenshot shows the same interface after solving the TSP problem. The "Status: Optimal" message is displayed in green. The "Custo total: 46.00" and "Rota ótima: A → E → D → C → B → A" are shown below. The "Tempo de resolução: 0.0890 segundos" is also indicated.

5. Problema de Cobertura (Set Covering)

- **Descrição:** Selecionar o menor número de conjuntos para cobrir todos os elementos.
- **Arquivo:** problema_cobertura.txt - Matriz de incidência entre elementos e conjuntos.
- **Formato:** CSV com matriz binária (elementos nas linhas, conjuntos nas colunas).

Screenshots:

Programação Linear Inteira (PLI)

A Programação Linear Inteira (PLI) resolve problemas exatos de otimização combinatória. Escolha abaixo qual problema deseja modelar e resolvê-lo.

Obs.: Removido por ora o "Modelo Genérico".

Selecionar o Problema de PLI a resolver:

Problema de Cobertura (Set Covering)

Formato: matriz de incidência binária (linhas = elementos, colunas = conjuntos).

Cargue matriz de cobertura (CSV)

Drag and drop file here
Limit 20MB per file + CSV.TXT

problem_cobertura.txt 127.0B

Unnamed_0	Corg1	Corg2	Corg3	Corg4	Corg5	Corg6
Element1	1	1	0	0	0	1
Element2	0	1	1	0	0	1
Element3	1	0	0	1	0	0
Element4	0	0	1	1	1	0
Element5	0	1	1	0	1	0

Resolver Set Covering

Status: Optimal

Cobertura mínima: 2

Conjuntos escolhidos:

- * [C12, C14]

Tempo de resolução: 0.1070 segundos

6. Problema de Facility Location

- **Descrição:** Decidir quais depósitos abrir e como atender clientes minimizando custos.
- **Arquivos:**
 - `facility_depositos.txt` - Custos fixos de abertura de cada depósito.
 - `facility_clientes.txt` - Custos de atendimento de clientes por cada depósito.
- **Formato:** CSVs especificando depósitos, clientes e custos.

Screenshots:

Facility Location

Cargue dois arquivos:

- Depósitos (facility_fixo)
- Clientes e custos (facility_cliente, custo)

CSV de depósitos

Depósitos:

facility	fixo
0 Deposit01	2.00
1 Deposit02	2.50
2 Deposit03	3.20
3 Deposit04	2.00

Clientes:

facility	cliente	custo
0 Deposit01	Cliente01	20
1 Deposit02	Cliente02	15
2 Deposit03	Cliente03	25
3 Deposit04	Cliente04	30
4 Deposit01	Cliente05	10
5 Deposit02	Cliente06	15
6 Deposit03	Cliente07	10
7 Deposit04	Cliente08	20
8 Deposit01	Cliente09	25
9 Deposit02	Cliente10	15

Resolver Facility Location

The screenshot shows a web-based application for solving combinatorial optimization problems using meta-heuristic methods. The main interface includes:

- Navegação (Navigation):** Features a logo with three red flames on top of a blue shield containing fish, and the motto "SCIENTIA AD SAPIENTIAM".
- Resolução da Instância (Resolution):** Displays a table of 10 entries with columns "Depósito" and "Cliente" and values ranging from 10 to 30.
- Status:** Shows "Status: Optimal" in green.
- Custo total (Total Cost):** Displays "480.00".
- Depósitos abertos (Open Depositories):** A JSON-like tree structure showing allocations from four clients to three different depots.
- Alocação cliente → depósito (Allocation Client → Depot):** A detailed view of the allocation mapping.
- Tempo de resolução (Resolution Time):** Shows "0.1070 segundos".

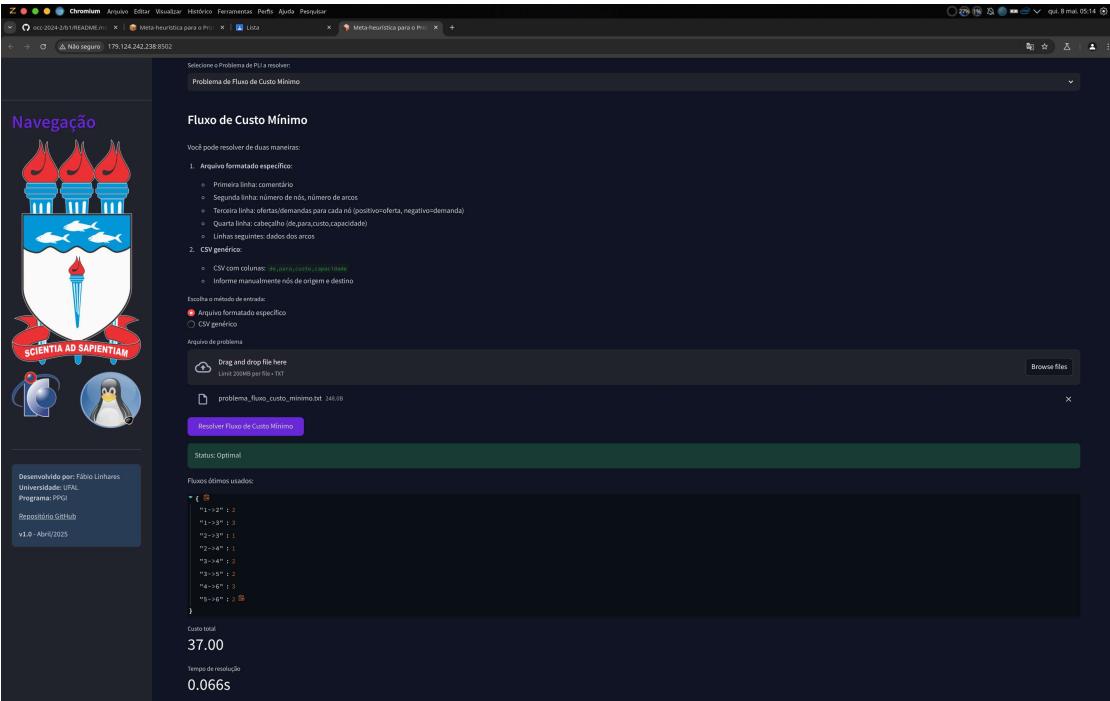
7. Problema de Fluxo de Custo Mínimo

- **Descrição:** Encontrar o fluxo de menor custo em uma rede.
- **Arquivo:** problema_fluxo_custo_minimo.txt - Arcos da rede com custos e capacidades.
- **Formato:** CSV com colunas: de, para, custo, capacidade.

Screenshots:

The screenshot shows the PLI application interface for solving integer linear programming problems. The specific section for the Minimum Cost Flow problem is displayed:

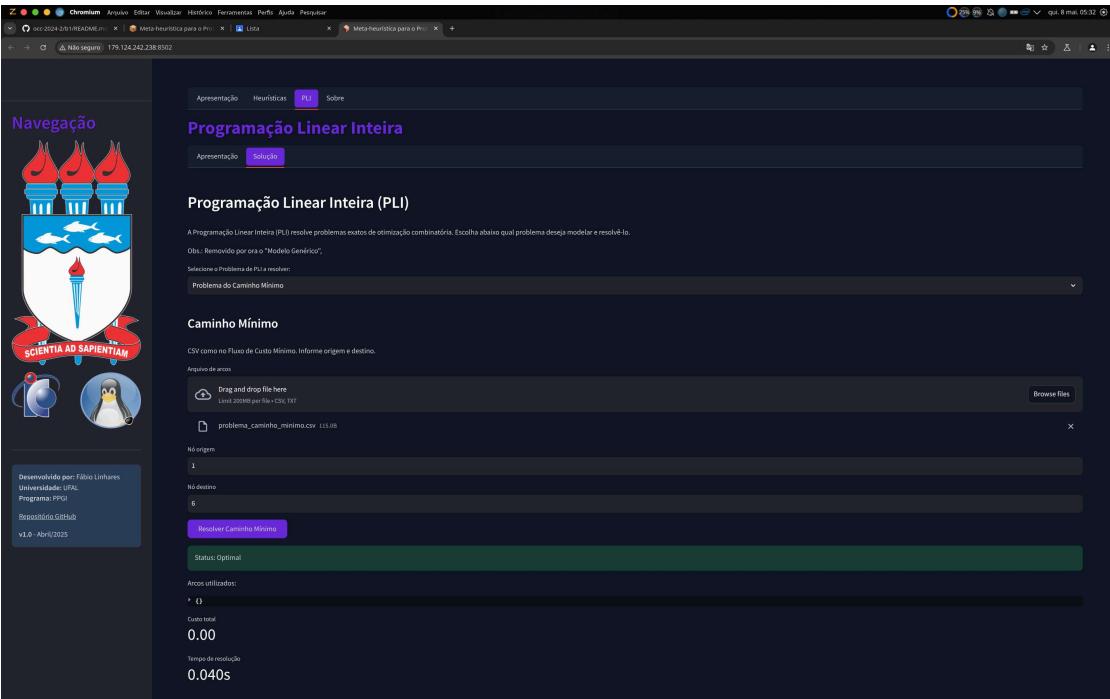
- Programação Linear Inteira (PLI):** Submenu "Solução" is selected.
- Problema de Fluxo de Custo Mínimo (Minimum Cost Flow Problem):** Descrição: "A Programação Linear Inteira (PLI) resolve problemas exatos de otimização combinatória. Escolha abaixo qual problema deseja modelar e resolvê-lo.".
- Arquivo de problema:** A file named "problema_fluxo_custo_minimo.txt" is selected for upload.
- Resoluver Fluxo de Custo Mínimo (Solve Minimum Cost Flow):** A button to start the resolution process.



8. Problema do Caminho Mínimo

- **Descrição:** Encontrar o caminho mais curto entre dois nós em um grafo.
- **Arquivo:** problema_caminho_minimo.txt - Arcos do grafo com custos.
- **Formato:** Similar ao fluxo de custo mínimo.

Screenshots:

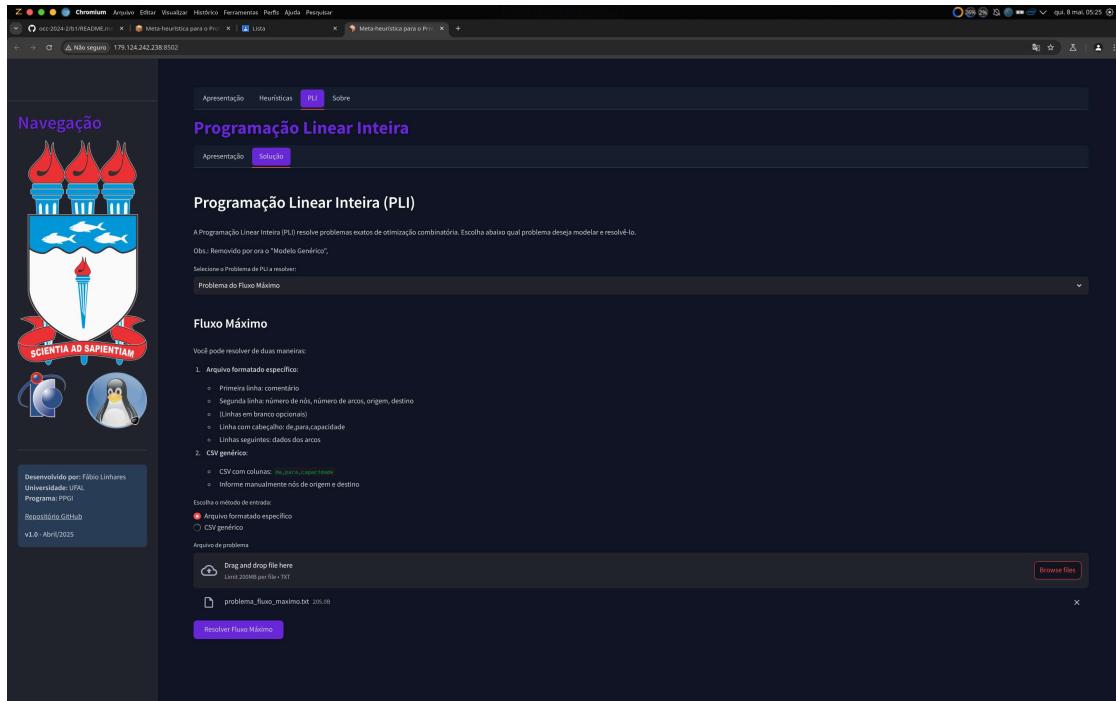


9. Problema do Fluxo Máximo

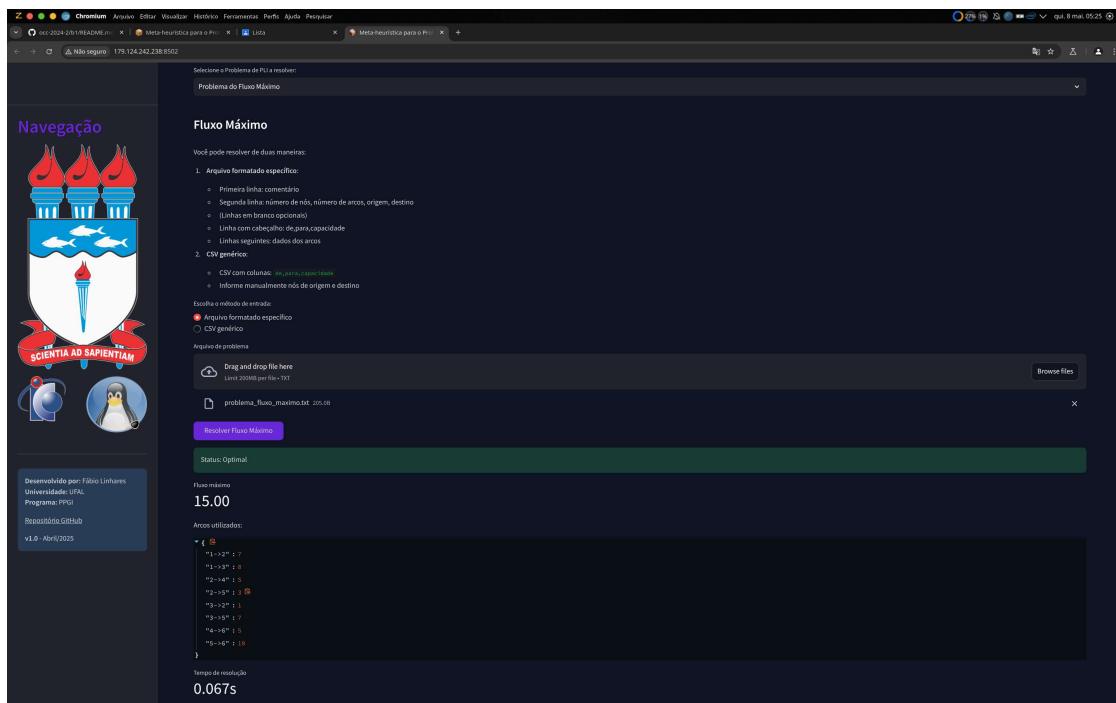
- **Descrição:** Maximizar o fluxo entre origem e destino em uma rede.
- **Arquivo:** problema_fluxo_maximo.txt - Arcos da rede com capacidades.

- **Formato:** CSV com colunas: de, para, capacidade.

Screenshots:



○

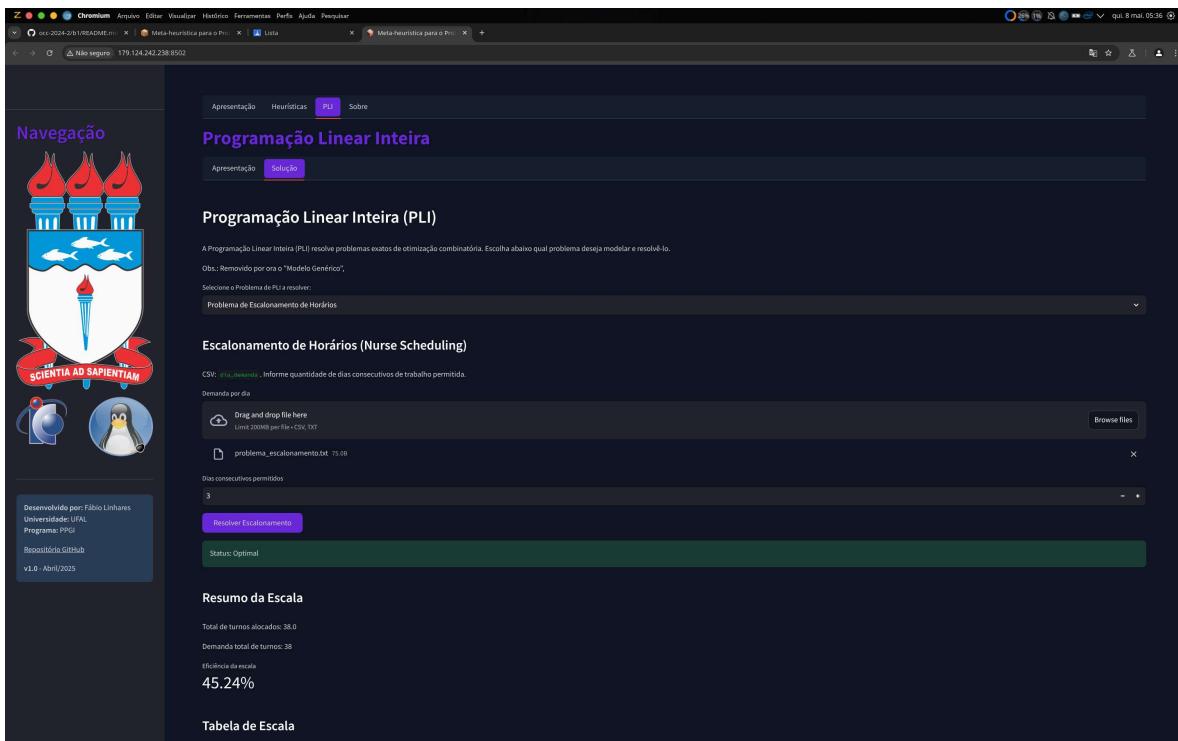


○

10. Problema de Escalonamento de Horários

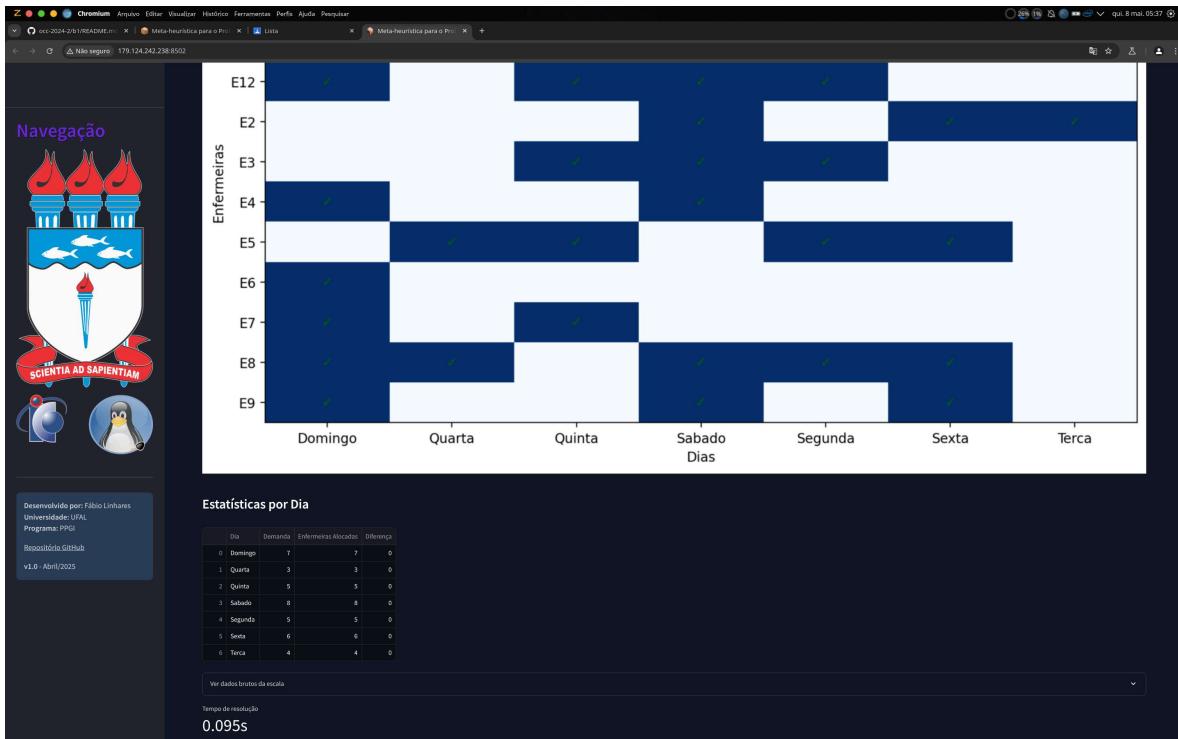
- **Descrição:** Alocar enfermeiras a turnos satisfazendo demandas e restrições.
- **Arquivo:** problema_escalonamento.txt - Demandas por dia.
- **Formato:** CSV com colunas: dia, demanda.

Screenshots:

● 

●  [Visualização da solução ótima](#)

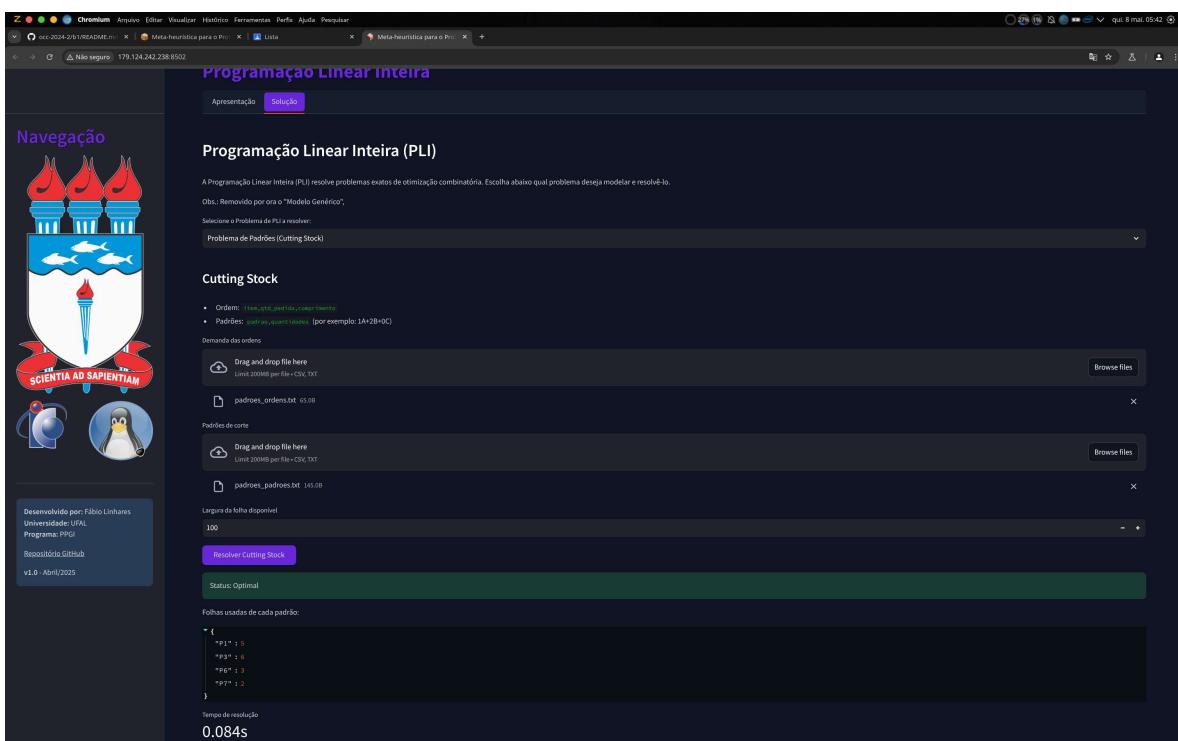




11. Problema de Padrões (Cutting Stock)

- **Descrição:** Cortar material minimizando o desperdício.
- **Arquivos:**
 - padroes_ordens.txt - Itens pedidos com quantidades e dimensões.
 - padroes_padroes.txt - Padrões de corte possíveis.
- **Formato:** CSVs especificando ordens e padrões de corte.

Screenshots:



12. Problema de Frequência (Frequency Assignment)

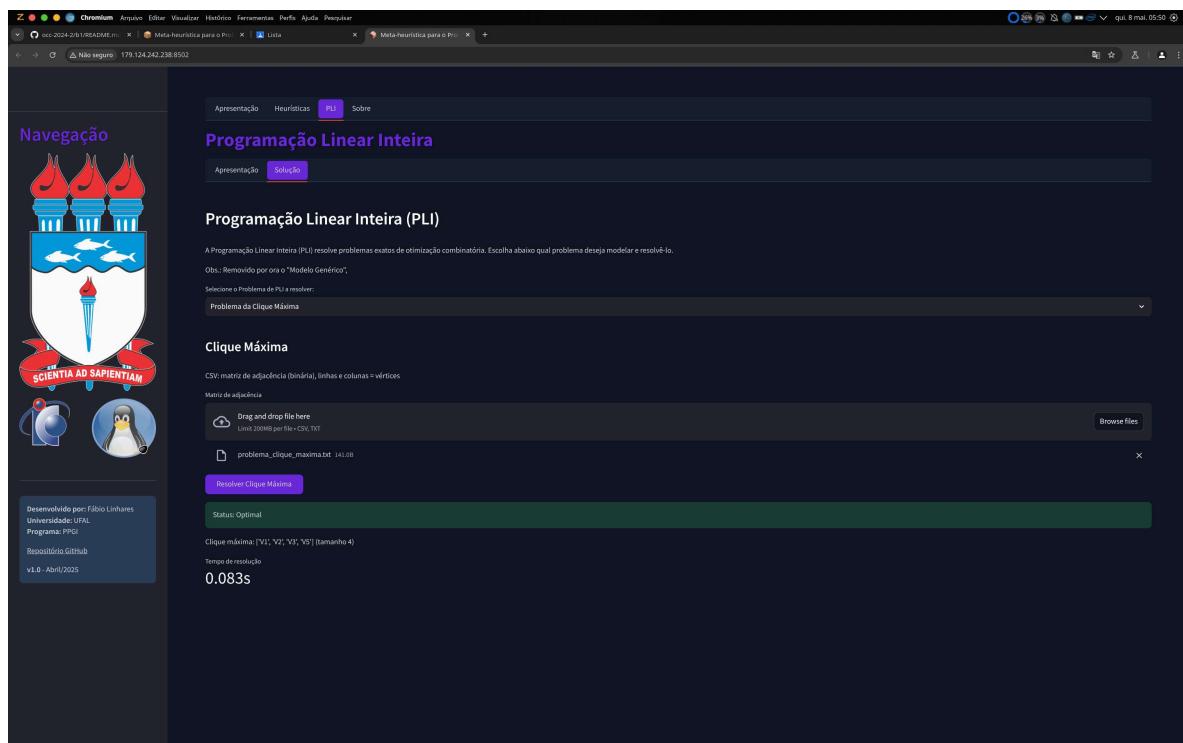
- **Descrição:** Atribuir frequências a antenas minimizando interferências.
- **Arquivos:**
 - `frequencia_antenas.txt` - Lista de antenas.
 - `frequencia_conflitos.txt` - Pares de antenas que interferem entre si.
- **Formato:** CSVs com antenas e conflitos.

Screenshots:

13. Problema da Clique Máxima

- **Descrição:** Encontrar o maior subconjunto de vértices completamente conectados em um grafo.
- **Arquivo:** `problema_clique_maxima.txt` - Matriz de adjacência do grafo.
- **Formato:** CSV com matriz binária de adjacência.

Screenshots:



14. Problema do Plantio

- **Descrição:** Determinar a área ótima de plantio para maximizar lucro com restrições.
- **Arquivos:**
 - `plantio_fazendas.txt` - Informações sobre fazendas.
 - `plantio_culturas.txt` - Informações sobre culturas.

- `plantio_restricoes.txt` - Restrições de proporção de culturas.
- **Formato:** CSVs com dados de fazendas, culturas e restrições.
- **Screenshots:**

Programação Linear Inteira (PLI)

A Programação Linear Inteira (PLI) resolve problemas exatos de otimização combinatoria. Escolha abaixo qual problema deseja modelar e resolvê-lo.

Obs.: Removido por ora o "Modelo Gênero".

Selecione o Problema de PLI a resolver:

Problema do Plantio

Plantio (Crop Planning)

- Fazendas: `fazenda_area.csv, agua_max.csv`
- Culturas: `cultura_lucro.csv, agua_por_area.csv`
- Restrições: `plantio_culturas.txt, plantio_restricoes.txt, proporcões_max.csv`

Fazendas

Culturas

Restrições

Fazendas: `fazenda_area.csv, agua_max.csv`

Culturas: `cultura_lucro.csv, agua_por_area.csv`

Restrições: `plantio_restricoes.txt, proporcões_max.csv`

Resolver Plantio

Plano de Plantio

Fazenda	Cultura	Área Plantada
0 Fazenda1	Algodão	35
0 Fazenda1	Milho	24
1 Fazenda1	Soja	5
2 Fazenda1	Trigo	25

Lucro total
R\$ 116300.00

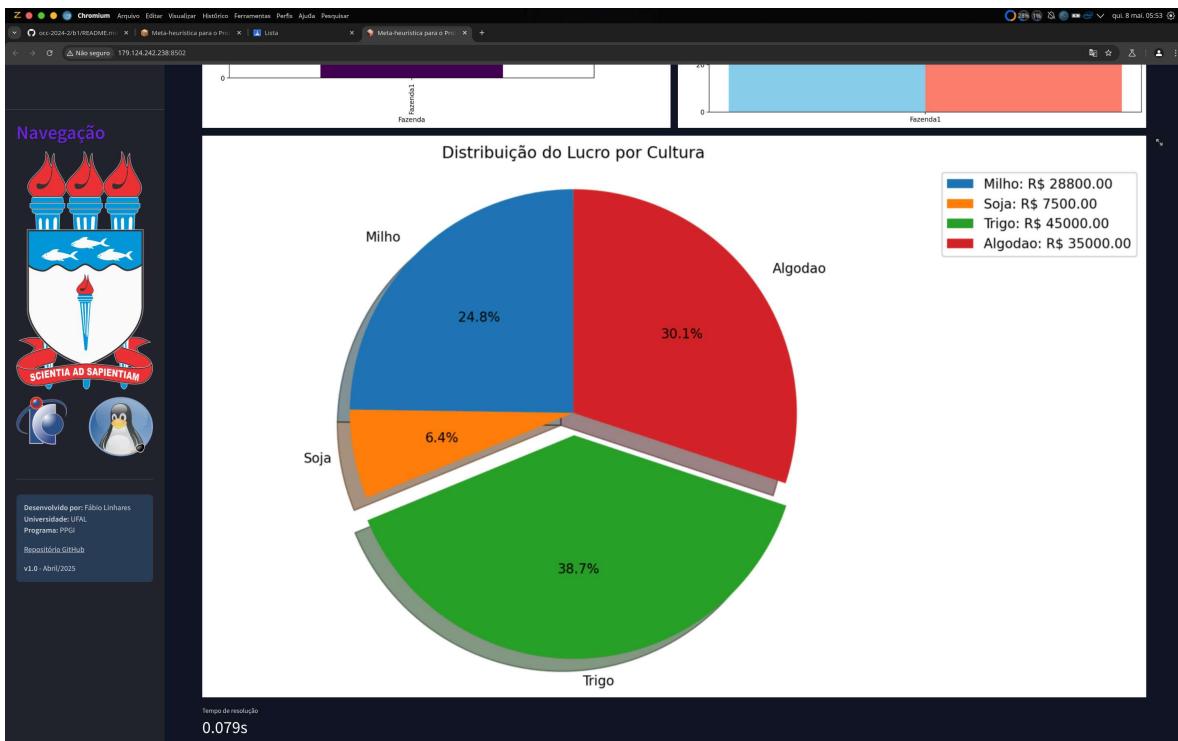
Estatísticas de Uso de Recursos

Fazenda	Área Total	Área Utilizada	Utilização de Área (%)	Água Disponível	Água Utilizada	Utilização de Água (%)
0 Fazenda1	100	89	89	450	450	100

Visualizações

Distribuição de Área por Fazenda

Utilização de Recursos por Fazenda



15. Problema das Tintas

- **Descrição:** Determinar a produção ótima de tintas com restrições de componentes.
- **Arquivos:**
 - `tintas_produtos.txt` - Produtos e preços de venda.
 - `tintas_componentes.txt` - Requisitos de componentes para cada produto.
 - `tintas_disponibilidade.txt` - Disponibilidade de componentes.
- **Formato:** CSVs com produtos, componentes e disponibilidade.

Screenshots:

Navegação

Programação Linear Inteira

Programação Linear Inteira (PLI)

A Programação Linear Inteira (PLI) resolve problemas exatos de otimização combinatória. Escolha abaixo qual problema deseja modelar e resolvê-lo.

Obs.: Removido por ora o "Modelo Gênero".

Selecione o Problema de PLI a resolver:

Problema das Tintas

Tintas (Blending Problem)

- Produtos: `produtos,preco_venda`
- Componentes: `produtos,componente,preco_venda`
- Disponibilidade: `componente,disponivel`

Produtos

Drag and drop file here
Limit 200MB per file - CSV, TXT

tintas_produtos.txt 53.08

Components

Drag and drop file here
Limit 200MB per file - CSV, TXT

tintas_componentes.txt 257.08

Disponibilidade

Drag and drop file here
Limit 200MB per file - CSV, TXT

tintas_disponibilidade.txt 78.08

Resolver Tintas

Navegação

Programação Linear Inteira

Programação Linear Inteira (PLI)

A Programação Linear Inteira (PLI) resolve problemas exatos de otimização combinatória. Escolha abaixo qual problema deseja modelar e resolvê-lo.

Obs.: Removido por ora o "Modelo Gênero".

Selecione o Problema de PLI a resolver:

Problema das Tintas

Tintas (Blending Problem)

- Produtos: `produtos,preco_venda`
- Componentes: `produtos,componente,preco_venda`
- Disponibilidade: `componente,disponivel`

Produtos

Drag and drop file here
Limit 200MB per file - CSV, TXT

tintas_produtos.txt 53.08

Components

Drag and drop file here
Limit 200MB per file - CSV, TXT

tintas_componentes.txt 257.08

Disponibilidade

Drag and drop file here
Limit 200MB per file - CSV, TXT

tintas_disponibilidade.txt 78.08

Resolver Tintas

Status: Optimal

Produção ótima:

```
{
    "TintaA": 28,
    "TintaB": 46,
    "TintaC": 71
}
```

Lucro total: 17360.00

Tempo de resolução: 0.088s

Como Executar

Requisitos

- Python 3.8+
- Streamlit
- Pandas
- NumPy
- Altair
- Matplotlib

- PIL (Pillow)
- PuLP (para os modelos ILP)

Instalação

```
# Clone o repositório  
git clone https://github.com/fabio-linhares/occ-2024-2.git  
  
# Crie um ambiente conda (opcional)  
conda create -n occ2024-2 python=3.12  
conda activate occ2024-2  
  
# Instale as dependências  
pip install streamlit pandas numpy altair matplotlib pillow pulp
```



Executando a Aplicação

Para iniciar a aplicação Streamlit, execute o seguinte comando no terminal:

```
streamlit run b1/app/main.py
```



O navegador será aberto automaticamente com a aplicação em execução. Se isso não acontecer, acesse:

- Local: <http://localhost:8501>
- Rede: <http://seu-ip:8501>

Utilizando a Aplicação

1. Navegue até a aba "Heurísticas" → "Solução"
2. Carregue um arquivo contendo uma instância do problema
 - o Cada linha do arquivo deve conter um número entre 0 e 1 (tamanho do item)
3. Configure o tempo limite de execução e o número máximo de bins
4. Clique em "Executar Meta-heurística"
5. Analise os resultados e visualizações

Estrutura do Projeto

```
b1/  
|   app/                      # Código da aplicação  
|   |   main.py                # Arquivo principal da aplicação  
|   |   ilp.py                 # Implementação dos modelos de PLI  
|   |   styles/                # Arquivos CSS para estilização
```



```
|   └── pages/          # Páginas da aplicação
├── data/              # Dados e exemplos
│   └── instances/      # Instâncias do problema
│       └── logos/        # Logos institucionais
└── README.md          # Este arquivo
```

Exemplos de Instâncias

O diretório `data/instances/` contém exemplos de instâncias do problema que podem ser utilizadas para testar a aplicação.

Desenvolvido por

Fábio Linhares

Universidade Federal de Alagoas (UFAL)

Programa de Pós-Graduação em Informática (PPGI)

[Repositório GitHub](#)