

# Algoritmos para r-Arborescências Geradoras Mínimas em Digrafos: Uma Aplicação Web Interativa

Lorena Sampaio, Samira Haddad

Orientador: Prof. Dr. Mário Leston Rey

Universidade Federal do ABC  
Centro de Matemática, Computação e Cognição

25 de novembro de 2025



# Sumário

- 1 Introdução
- 2 Algoritmo de Chu-Liu-Edmonds
- 3 Algoritmo de András Frank
- 4 Resultados Experimentais
- 5 Aplicação Web
- 6 Conclusões



# O Problema

Encontrar uma  $r$ -Arborescência Geradora de Custo Mínimo

Dado um  $r$ -digafo ponderado  $(D, w, r)$ :

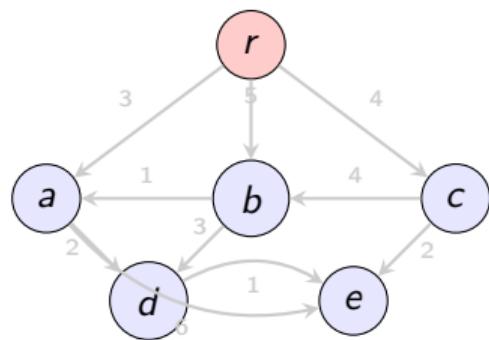
- Encontrar uma  $r$ -arborescência geradora de custo mínimo de  $D$

**Algoritmos estudados:**

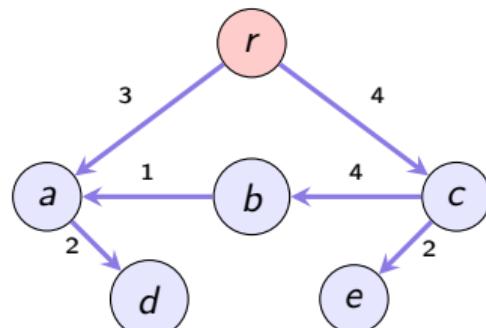
- ① Chu-Liu-Edmonds (1965-67)
- ② András Frank (1981-2014)

# Exemplo: $r$ -Arborescência Geradora

Exemplo de uma  $r$ -arborescência mínima:



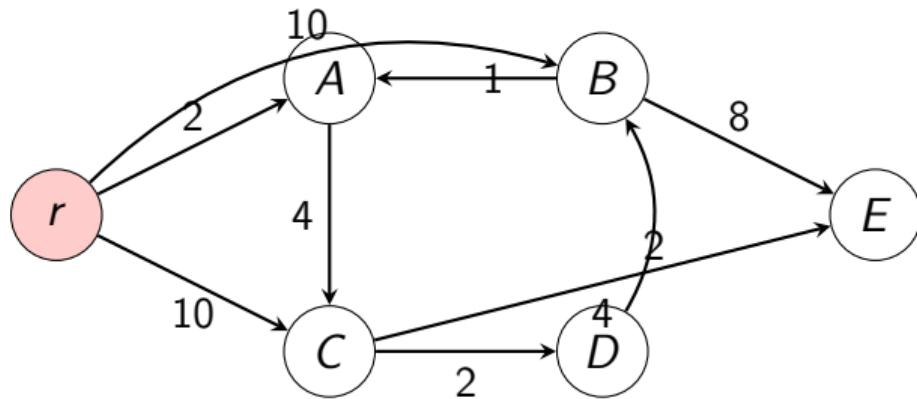
Digrafo Original



$r$ -Arborescência Geradora Mínima

$$\text{Custo total: } 3 + 4 + 1 + 2 + 2 + 4 = 16$$

# Digrafo de Exemplo



## Objetivo

Encontrar a  $r$ -arborescência geradora de custo mínimo

# Chu-Liu-Edmonds: Ideia Principal

## Estratégia Recursiva

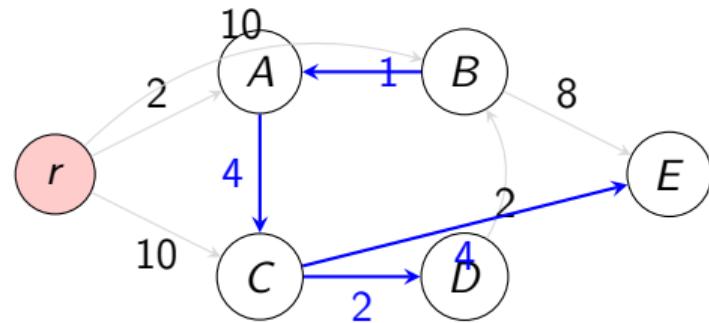
- ① Para cada vértice  $v \neq r$ : escolher arco de entrada de **custo mínimo**
- ② Se não há ciclos  $\Rightarrow$  solução ótima encontrada!
- ③ Se há ciclos  $\Rightarrow$  **contrair** e resolver recursivamente

**Escolha Gulosa + Contração de Ciclos**

# Passo 1: Escolha Gulosa

Para cada vértice, selecionar arco de entrada mínimo:

- A: arco  $(B, A)$  peso 1 ✓
- B: arco  $(r, B)$  peso 10
- C: arco  $(A, C)$  peso 4 ✓
- D: arco  $(C, D)$  peso 2 ✓
- E: arco  $(C, E)$  peso 4 ✓



## Problema

Ciclo detectado:  $A \rightarrow C \rightarrow D \rightarrow B \rightarrow A$

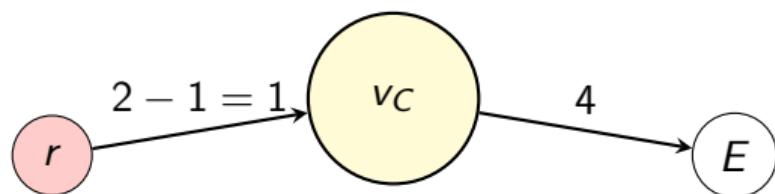
## Passo 2: Contração do Ciclo

**Ciclo encontrado:**  $\{A, B, C, D\}$

Contraímos em um super-vértice  $v_C$

Ajustamos pesos dos arcos que **entram** no ciclo:

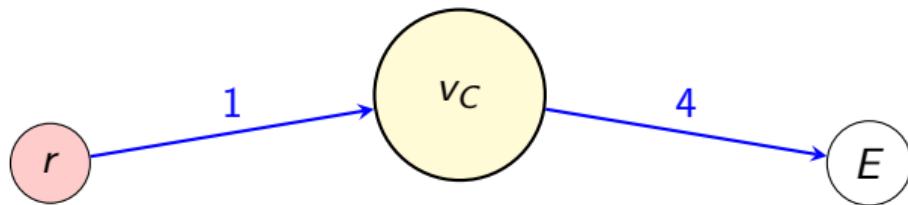
$$w'(u, v_C) = w(u, v) - w(\min\_in(v))$$



### Recursão

Resolver o problema no digrafo contraído

## Passo 3: Solução no Digrafo Contraído



Escolha gulosa:

- $(r, v_C)$  peso 1
- $(v_C, E)$  peso 4

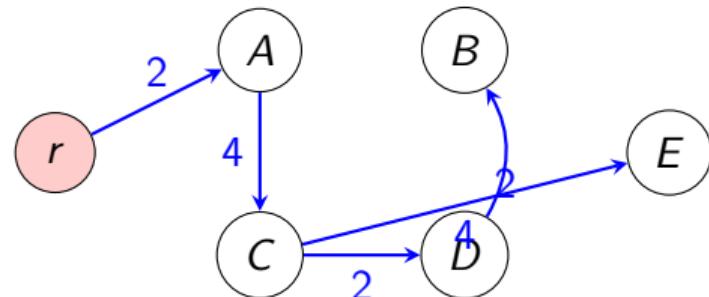
Sem ciclos!  $\Rightarrow$  Solução encontrada no digrafo contraído

## Passo 4: Expandindo a Solução

No digrafo contraído:  $(r, v_C)$  foi escolhido

Expandindo  $v_C$ :

- Remover arco  $(B, A)$  do ciclo
- Manter:  $(A, C), (C, D), (D, B)$
- Adicionar:  $(r, A)$  peso 2



### Solução Ótima

Custo total:  $2 + 4 + 2 + 2 + 4 = 14$

# András Frank: Visão Geral

## Abordagem em Duas Fases

**Fase I:** Construir cobertura de subconjuntos minimais via redução de custos

**Fase II:** Extrair arborescência da cobertura

### Diferencial:

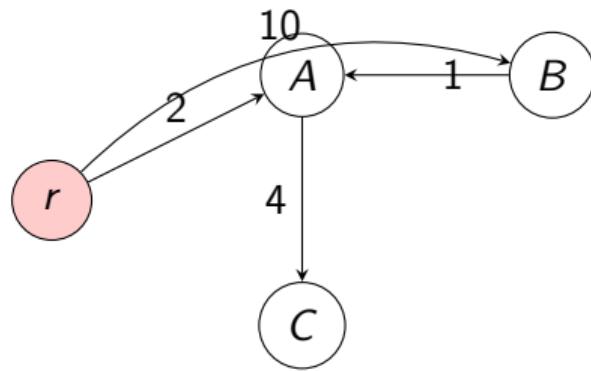
- Trabalha com múltiplos vértices simultaneamente
- Usa componentes fortemente conexas
- Redução sistemática de custos

### Complexidade:

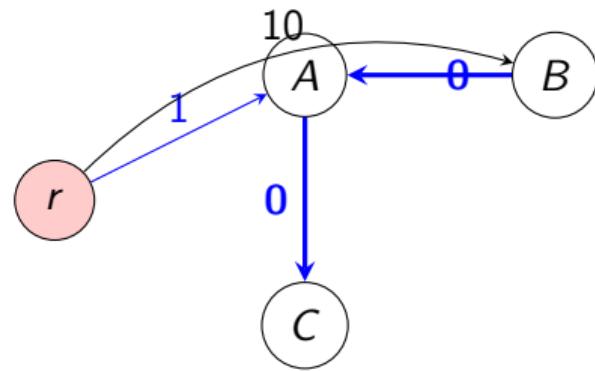
- Fase I:  $O(nm)$
- Fase II v1 (lista):  $O(n^2)$
- Fase II v2 (heap):  $O(n \log n)$

# Fase I: Redução de Custos

Para cada vértice  $v \neq r$ : subtrair o mínimo de entrada



Original



Após Redução

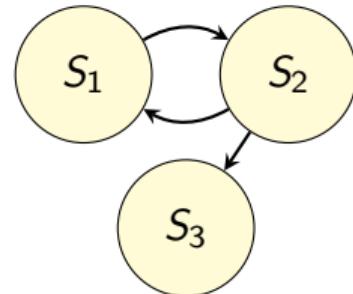
Arcos com custo zero formam o digrafo  $D_0$

# Fase I: Componentes Fortemente Conexas

Identificar componentes fortemente conexas (CFCs) em  $D_0$

Cada CFC forma um **subconjunto minimal**

Construir sequência laminar de subconjuntos



## Condição de Optimalidade

Sequência  $\lambda$  satisfaz:  $|\delta^-(X)| = 1$  para cada  $X$  em  $\lambda$

# Fase II: Construção da Arborescência

**Objetivo:** Extrair arborescência de  $D_0$  respeitando  $\lambda$

- ① Iniciar com conjunto  $R = \{r\}$
- ② Para cada  $v$  fora de  $R$ :
  - Selecionar arco  $(u, v)$  com  $u \in R$  e custo reduzido zero
  - Adicionar  $v$  a  $R$
- ③ Repetir até incluir todos os vértices

## Resultado

Arborescência ótima com mesma solução: custo 14

# Comparação de Desempenho

Experimentos: 2000 digrafos aleatórios,  $|V| \in [101, 4996]$

Algoritmo	Tempo Mediano	Tempo Médio
Chu-Liu-Edmonds	0,25 s	0,58 s
Frank Fase I	8,93 s	12,40 s
Frank Fase II (lista)	0,98 s	1,34 s
Frank Fase II (heap)	<b>0,016 s</b>	<b>0,020 s</b>

## Speedup Fase II

Heap vs Lista: aceleração de 58,12 vezes (mediana)

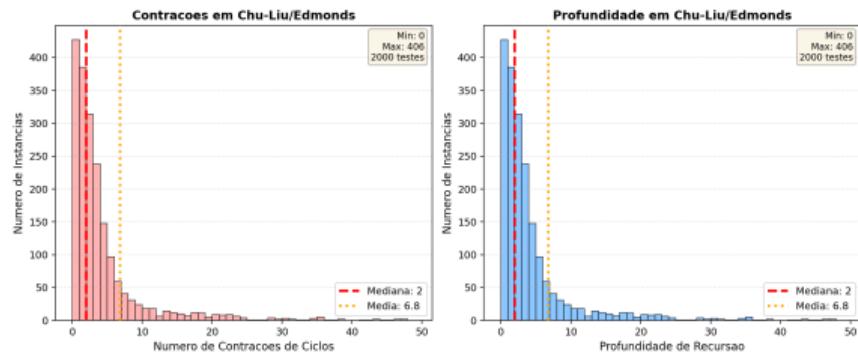
# Características Estruturais

## Contrações (Chu-Liu):

- Mediana: 2 contrações
- Média: 6,82
- Máximo: 406
- 93,8% com < 20

Muito abaixo do limite teórico  $O(n)$

**Consumo de memória:** mediana 11,5 MB (Fase I)



# Motivação Didática

## Desafio

Algoritmos de grafos são **abstratos e difíceis de visualizar**

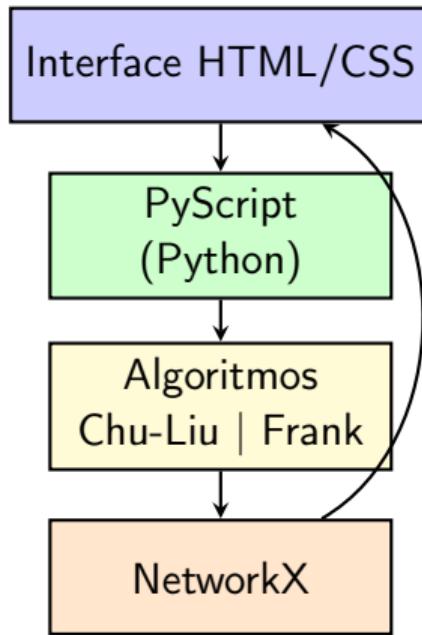
## Solução Proposta:

- Visualização interativa
- Execução passo a passo
- Feedback imediato
- Acessível via navegador

## Tecnologias:

- PyScript (Python no browser)
- JavaScript
- HTML5/CSS3
- NetworkX

# Arquitetura da Aplicação



# Interface: Página Principal

 ArboGraph

Algoritmos para o problema da arborescência geradora mínima: uma aplicação didática interativa

- Home
- Chu-Liu-Edmonds
- András Frank (V1)
- András Frank (V2)
- Desenhe um digrafo
- Nossa dissertação

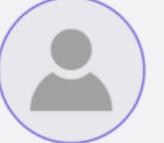
Dúvidas ?  
Quer aprender mais sobre esses algoritmos, leia nossa tese :)

[Link](#)

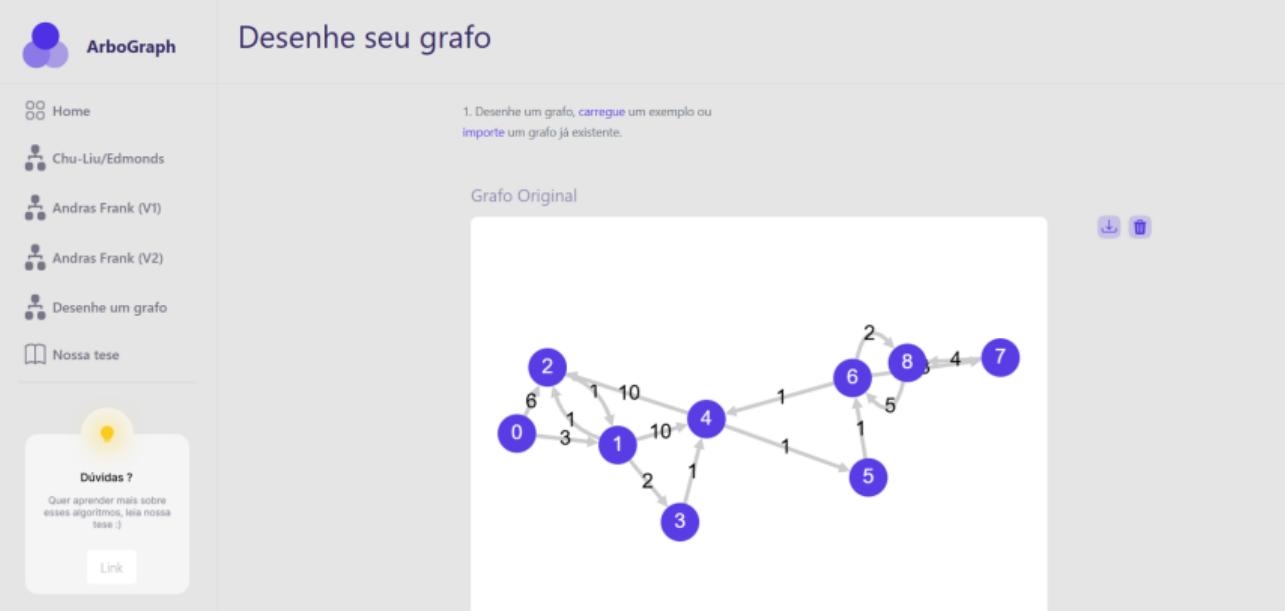
**Resumo**

Este trabalho investiga e implementa algoritmos de busca de uma  $r$ -arborescência geradora mínima em digrafos. A partir da formulação clássica e da literatura de Chu-Liu-Edmonds e também da formulação de András Frank, desenvolvemos uma aplicação web que permite: (i) desenhar ou importar um digrafo ponderado, (ii) escolher o nó raiz  $r$ , (iii) executar o algoritmo passo a passo com visualização das contrações, seleção de arcos de custo mínimo e reconstrução da arborescência, e (iv) exportar resultados e logs. A solução combina PyScript e NetworkX para a lógica algorítmica, Cytoscape para edição e visualização interativa, e Tailwind/Flowbite na interface. Como contribuição, o sistema oferece um ambiente didático que torna transparentes as decisões do algoritmo e facilita a análise e comparação de soluções em diferentes instâncias, apoiando ensino, experimentação e validação.

Integrantes do Projeto



# Interface: Desenho de Grafos



The screenshot shows the ArboGraph web application interface. On the left, there is a sidebar with the following menu items:

- Home
- Chu-Liu/Edmonds
- Andras Frank (V1)
- Andras Frank (V2)
- Desenhe um grafo
- Nossa tese

Below the menu is a yellow button labeled "Dúvidas ?" with a tooltip: "Quer aprender mais sobre esses algoritmos, leia nossa tese :)" and a "Link" button.

The main area is titled "Desenhe seu grafo" and contains instructions: "1. Desenhe um grafo, carregue um exemplo ou importe um grafo já existente." Below this is a section titled "Grafo Original" showing a directed graph with 9 nodes (0-8) and edges with weights:

```

graph LR
    0((0)) -- 3 --> 1((1))
    0((0)) -- 6 --> 2((2))
    1((1)) -- 1 --> 0((0))
    1((1)) -- 10 --> 4((4))
    1((1)) -- 2 --> 3((3))
    2((2)) -- 1 --> 1((1))
    3((3)) -- 1 --> 4((4))
    4((4)) -- 10 --> 1((1))
    4((4)) -- 1 --> 5((5))
    4((4)) -- 1 --> 6((6))
    5((5)) -- 1 --> 4((4))
    5((5)) -- 1 --> 6((6))
    5((5)) -- 1 --> 7((7))
    6((6)) -- 2 --> 5((5))
    6((6)) -- 5 --> 8((8))
    6((6)) -- 4 --> 7((7))
    7((7)) -- 2 --> 6((6))
    8((8)) -- 4 --> 7((7))
  
```

On the right side of the graph area are two small icons: a downward arrow and a trash can.

## Funcionalidades:

- Adicionar vértices e arestas
- Definir pesos

# Interface: Chu-Liu-Edmonds



**ArboGraph**

- Home
- Chu-Liu/Edmonds
- ▢▢ Andras Frank (V1)
- ▢▢ Andras Frank (V2)
- ▢▢ Desenhe um grafo
- ▢▢ Nossa tese

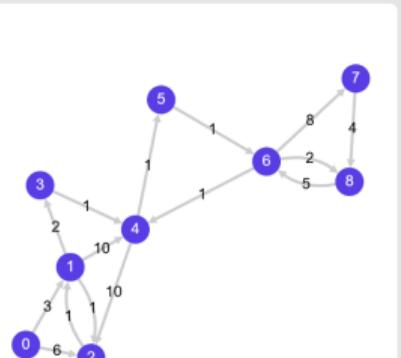
**Dúvidas ?**  
Quer aprender mais sobre esses algoritmos, leia nossa tese :)

[Link](#)

## Chu-Liu / Edmonds

1 Crie um grafo  
Desenhe um grafo, [carregue](#) um exemplo ou [importe](#) um grafo já existente.
2 Escolha o nó raiz  
3 Execute o algoritmo  
[»](#)

**Grafo Original**



The graph consists of 9 nodes labeled 0 through 8. Node 0 is the root node. There are directed edges between nodes 0 and 1 (weight 6), 0 and 2 (weight 1), 1 and 3 (weight 2), 1 and 4 (weight 10), 2 and 1 (weight 3), 2 and 4 (weight 1), 3 and 1 (weight 1), 4 and 5 (weight 1), 4 and 6 (weight 1), 5 and 6 (weight 1), 6 and 7 (weight 8), 6 and 8 (weight 2), 7 and 8 (weight 4), and 8 and 6 (weight 5).

» Passo A Passo

Execute o algoritmo para visualizar o passo-a-passo

- Visualização passo a passo
- Destacamento de ciclos detectados
- Log detalhado das operações

# Interface: András Frank

The screenshot shows the ArboGraph web application interface for version 1. The left sidebar contains a logo and links to Home, Chu-Liu/Edmonds, András Frank (V1), András Frank (V2), Draw a graph, and Our thesis. A 'Dúvidas?' (Questions?) button is also present. The main area is titled 'Andras Frank v1' and contains three numbered steps: 1. Create a graph (instructions: draw a graph, upload an example, or import an existing one), 2. Choose the root node (input field labeled 'node'), and 3. Run the algorithm (button with a play icon). To the right of step 3 is a 'Passo A Passo' (Step-by-step) section with a button to execute the algorithm. Below the steps is a placeholder for the 'Grafo Original' (Original Graph).

- Exibição das duas fases
- Visualização de CFCs
- Comparação entre versões (lista vs heap)

# Princípios de Design

## Teoria dos Registros de Representação (Duval)

Transitar entre diferentes representações:

- **Visual:** diagramas do grafo
- **Simbólico:** código Python
- **Textual:** log das operações

## Feedback Imediato

Validação em tempo real das operações do usuário



UFABC

# Contribuições do Trabalho

## ① Implementação completa de dois algoritmos clássicos

- Chu-Liu-Edmonds: recursivo com contração
- András Frank: duas fases com otimização heap

## ② Análise experimental detalhada

- 2000 instâncias aleatórias
- Comparação de desempenho e características estruturais

## ③ Aplicação web interativa

- Ferramenta didática para visualização
- Execução passo a passo dos algoritmos
- Design centrado no usuário

# Principais Resultados

- **Corretude validada:** custos idênticos em todas as instâncias
- **Chu-Liu-Edmonds** mais rápido para construção direta
  - Mediana: 0,25 s vs 8,93 s (Fase I Frank)
- **Otimização heap fundamental na Fase II**
  - Speedup: 58× (mediana), 61× (média)
- **Comportamento prático** muito melhor que limites teóricos
  - Contrações: mediana 2 (limite  $O(n)$ )
  - Memória modesta: 11,5 MB

# Trabalhos Futuros

## Extensões Possíveis

- Implementar outras variantes (Tarjan, Gabow)
- Análise em grafos com estruturas especiais
- Paralelização dos algoritmos
- Extensão para grafos dinâmicos

## Melhorias na Aplicação

- Modo de edição visual de grafos
- Geração automática de casos de teste
- Exercícios interativos com correção automática
- Integração com plataformas de ensino (Moodle, Jupyter)

# Obrigado!

Perguntas?

<https://github.com/lorenypsum/graph-visualizer>