

# Algoritmos para $r$ -Arborescências Geradoras Mínimas em Digrafos

## Aplicação Web Interativa

Lorena Sampaio | Samira Haddad

Universidade Federal do ABC  
Instituto de Matemática e Estatística

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

## Arborescência de Custo Mínimo

Dado um digrafo  $D = (V, A)$  com pesos  $w : A \rightarrow \mathbb{R}$  e raiz  $r \in V$ :

- Encontrar uma  $r$ -arborescência geradora  $T$
- Que minimize  $w(T) = \sum_{a \in A(T)} w(a)$

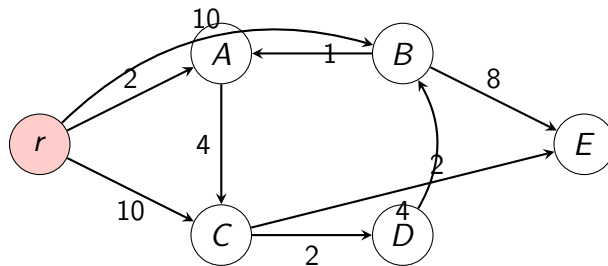
### Aplicações:

- Redes de comunicação
- Logística
- Otimização de recursos

### Algoritmos estudados:

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

# Digrafo de Exemplo



## Objetivo

Encontrar a  $r$ -arborescência geradora de custo mínimo

## Estratégia Recursiva

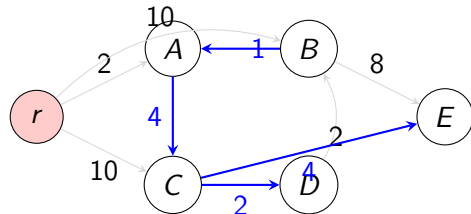
- 1 Para cada vértice  $v \neq r$ : escolher arco de entrada de **custo mínimo**
- 2 Se não há ciclos  $\Rightarrow$  solução ótima encontrada!
- 3 Se há ciclos  $\Rightarrow$  **contrair** e resolver recursivamente

**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:  $A \rightarrow C \rightarrow D \rightarrow B \rightarrow A$

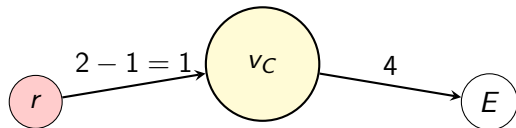
## Passo 2: Contração do Ciclo

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

Contraímos em um super-vértice  $v_C$

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

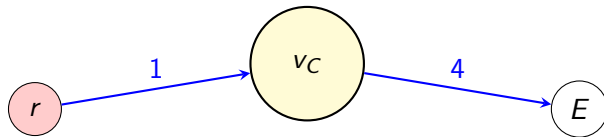
$$w'(u, v_C) = w(u, v) - w(\min\_in(v))$$



### 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

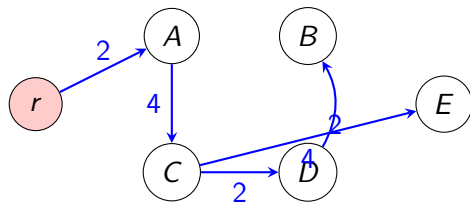


## Passo 4: Expandindo a Solução

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:  $2 + 4 + 2 + 2 + 4 = 14$

## Abordagem em Duas Fases

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

**Fase II:** Extrair arborescência da cobertura

### Diferencial:

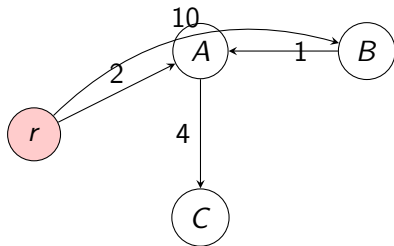
- Trabalha com múltiplos vértices simultaneamente
- Usa componentes fortemente conexas
- Redução sistemática de custos

### Complexidade:

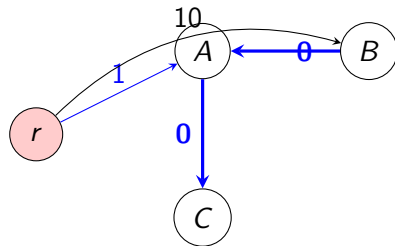
- Fase I:  $O(nm)$
- Fase II v1 (lista):  $O(n^2)$
- Fase II v2 (heap):  $O(n \log n)$

# Fase I: Redução de Custos

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



Original



Após Redução

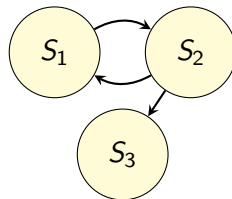
Arcos com custo **zero** formam o digrafo  $D_0$

# 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 Otimalidade

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

## Fase II: Construção da Arborescência

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

- ❶ Iniciar com conjunto  $R = \{r\}$
- ❷ Para cada  $v$  fora de  $R$ :
  - Selecionar arco  $(u, v)$  com  $u \in R$  e custo reduzido zero
  - Adicionar  $v$  a  $R$
- ❸ 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

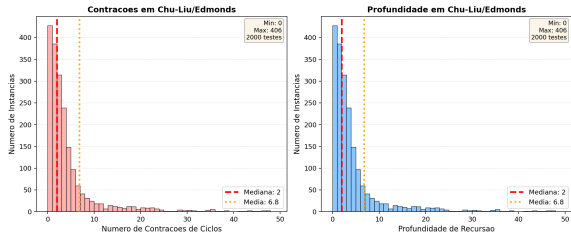
Heap vs Lista: aceleração de **58,12 vezes** (mediana)

## 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)



## Desafio

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

### Solução Proposta:

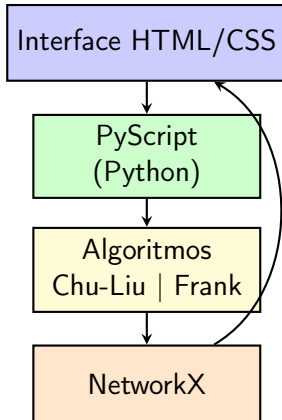
- Visualização interativa
- Execução passo a passo
- Feedback imediato
- Acessível via navegador

### Tecnologias:

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



# Arquitetura da Aplicação



# Interface: Página Principal

**ArboGraph**

 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](#)

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

### 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

**Lori Sampaio**  
Discente do BCC-UEABC

**Mario Leston Rey**  
Docente do BCC-UEABC

**Samira Haddad**  
Discente do BCC-UEABC


# Interface: Desenho de Grafos

The screenshot displays the ArboGraph web interface. On the left is a sidebar with a logo and a menu containing: Home, Chu-Liu/Edmonds, Andras Frank (V1), Andras Frank (V2), Desenhe um grafo (highlighted), and Nossa tese. Below the menu is a 'Dúvidas?' section with a lightbulb icon 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 'Grafo Original' section showing a directed graph with 9 nodes (0-8) and weighted edges. The graph structure is as follows: Node 0 points to Node 2 (weight 6) and Node 1 (weight 3). Node 2 points to Node 1 (weight 1) and Node 4 (weight 10). Node 1 points to Node 3 (weight 2) and Node 4 (weight 10). Node 3 points to Node 4 (weight 1). Node 4 points to Node 5 (weight 1) and Node 6 (weight 1). Node 5 points to Node 6 (weight 1). Node 6 points to Node 8 (weight 2) and Node 7 (weight 4). Node 8 points to Node 7 (weight 3). There are also self-loops on nodes 2, 4, 6, and 8. To the right of the graph are download and delete icons.

## Funcionalidades:

- Adicionar vértices e arestas
- Definir pesos
- Exportar/importar grafos em ISON

# 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

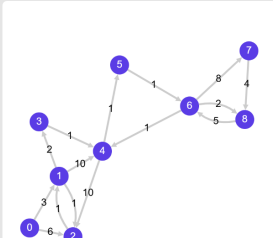
## Chu-Liu / Edmonds



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



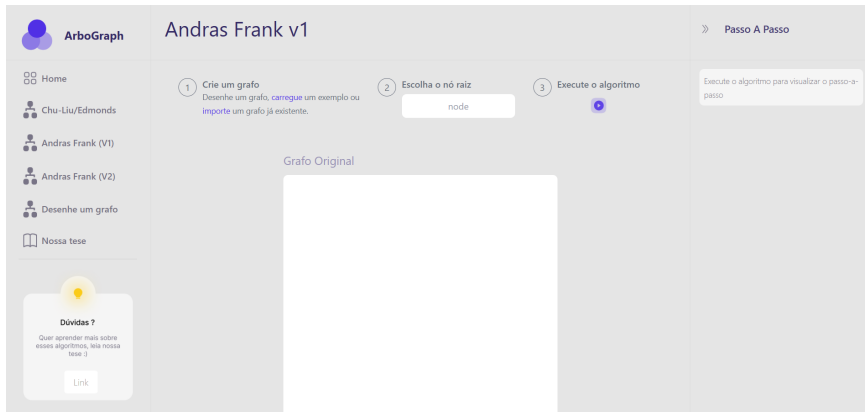
 

» Passo A Passo

Execute o algoritmo para visualizar o passo-a-passo

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

# Interface: András Frank



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

## 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

- ❶ **Implementação completa** de dois algoritmos clássicos
  - Chu-Liu-Edmonds: recursivo com contração
  - András Frank: duas fases com otimização heap
- ❷ **Análise experimental** detalhada
  - 2000 instâncias aleatórias
  - Comparação de desempenho e características estruturais
- ❸ **Aplicação web interativa**
  - Ferramenta didática para visualização
  - Execução passo a passo dos algoritmos
  - Design centrado no usuário

- **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:  $58\times$  (mediana),  $61\times$  (média)
- **Comportamento prático** muito melhor que limites teóricos
  - Contrações: mediana 2 (limite  $O(n)$ )
  - Memória modesta: 11,5 MB



## 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>