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

UFABC

Sumário

UFABC

O Problema

Encontrar uma r-Arborescência Geradora de Custo Mínimo

Dado um r-digrafo ponderado (D, w, r):

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

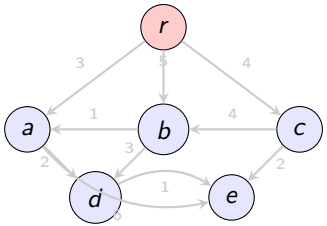
Algoritmos estudados:

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

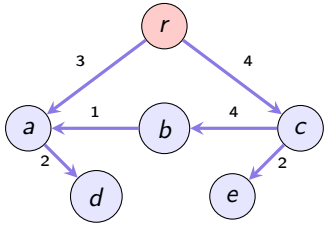
UFABC

Exemplo: r-Arborescência Geradora

Exemplo de uma r-arborescência mínima:



Digrafo Original

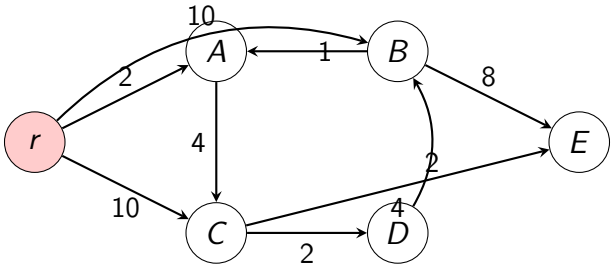


r-Arborescência Geradora Mínima

Custo total: $3 + 4 + 1 + 2 + 2 + 4 = 16$

UFABC

Digrafo de Exemplo



Objetivo

Encontrar a r-arborescência geradora de custo mínimo

UFABC

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

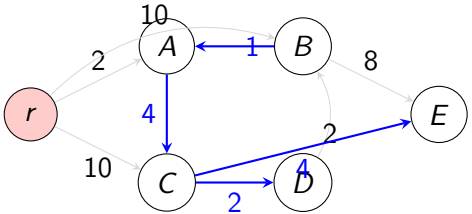
Escolha Gulosa + Contração de Ciclos

UFABC

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: $A \to C \to D \to B \to A$

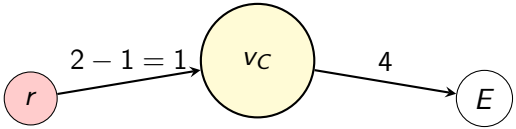
Passo 2: Contração do Ciclo

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

Contraímos em um super-vértice v_C

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

$$w'(u, v_C) = w(u, v) - w(\min\_\text{in}(v))$$

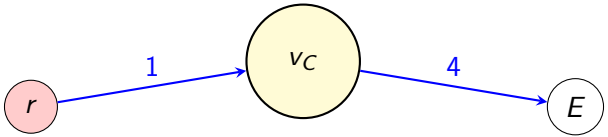


Recursão

Resolver o problema no digrafo contraído

UFABC

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

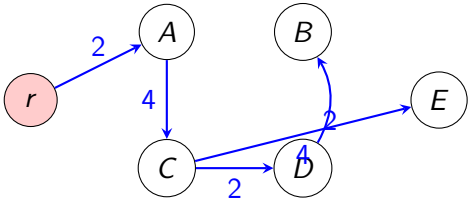
UFABC

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

UFABC

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

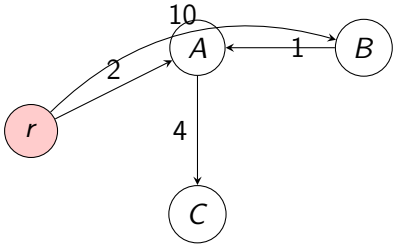
Complexidade:

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

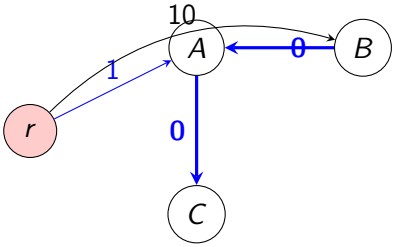
UFABC

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

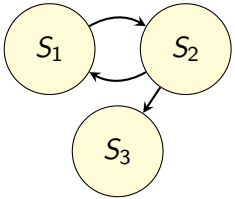
UFABC

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

UFABC

Fase II: Construção da Arborescência

Objetivo: Extrair arborescência de D_0 respeitando λ

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

UFABC

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)	**0,016 s**	**0,020 s**

Speedup Fase II

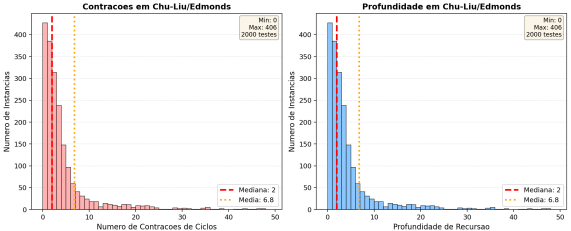
Heap vs Lista: aceleração de **58,12 vezes** (mediana)

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)

UFABC

Motivação Didática

Desafio

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

Solução Proposta:

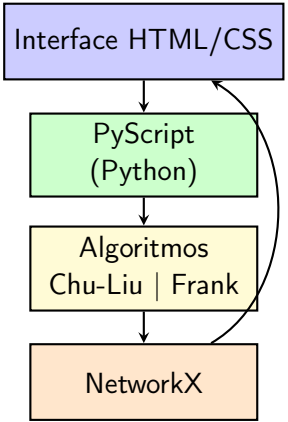
- Visualização interativa
- Execução passo a passo
- Feedback imediato
- Acessível via navegador

Tecnologias:

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

UFABC

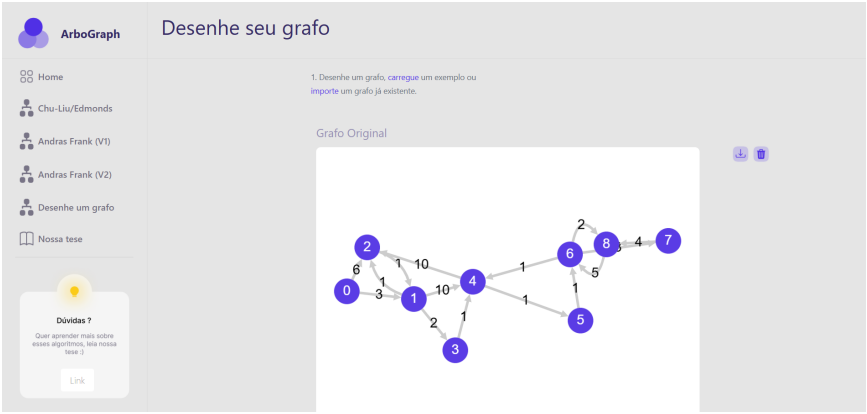
Arquitetura da Aplicação



UFABC

Interface: Página Principal

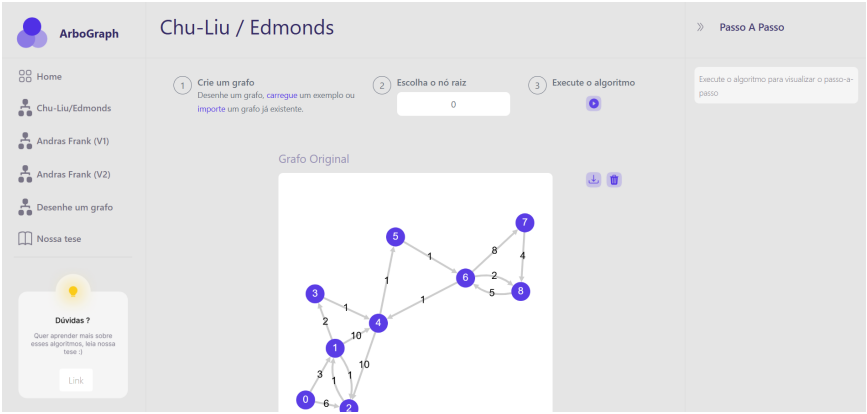
Interface: Desenho de Grafos



Funcionalidades:

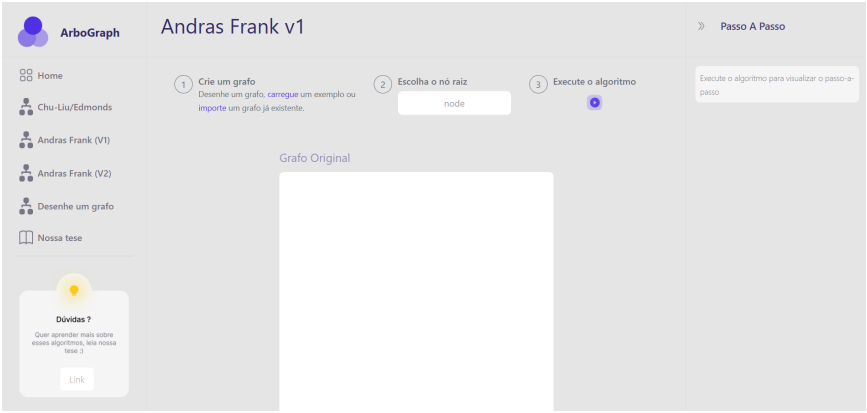
- Adicionar vértices e arestas
- Definir pesos

Interface: Chu-Liu-Edmonds



- 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)

UFABC

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

UFABC

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

UFABC

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`