

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 -digrafo ponderado (D, w, r) :

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

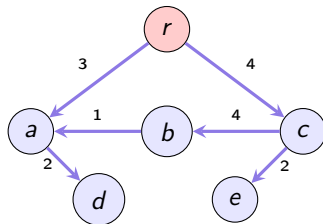
Algoritmos estudados:

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

Exemplo: r -Arborescência Geradora Mínima



Digrafo Original

 r -Arborescência Geradora

Custo: 16



Geradora Mínima

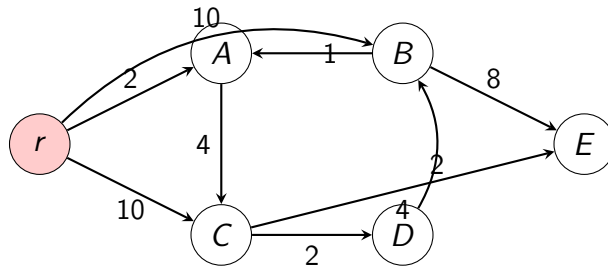
Custo: 13

Chu-Liu-Edmonds: Ideia Principal

Algoritmo Recursivo

- 1 **Reduzir custos:** para cada vértice $v \neq r$, subtrair $\lambda(v) = \min\{w(a) : a \in \delta^-(v)\}$
Em outras palavras: tornar zero o custo do arco mais barato que entra em cada vértice
- 2 **Construir D_0 :** escolher um arco a_v de custo reduzido zero para cada $v \neq r$
- 3 **Verificar:** se D_0 é uma r -arborescência \Rightarrow **devolver D_0**
- 4 **Caso contrário:** encontrar ciclo C em D_0 , contrair e resolver recursivamente

Digrafo de Exemplo



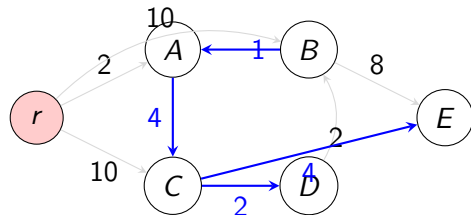
Objetivo

Encontrar a r -arborescência geradora de custo mínimo

Passo 1: Redução de Custos e Construção de D_0

Redução de custos λ :

- $\lambda(A) = 1 \Rightarrow (B, A): 1 - 1 = 0$
- $\lambda(B) = 2 \Rightarrow (D, B): 2 - 2 = 0$
- $\lambda(C) = 4 \Rightarrow (A, C): 4 - 4 = 0$
- $\lambda(D) = 2 \Rightarrow (C, D): 2 - 2 = 0$
- $\lambda(E) = 4 \Rightarrow (C, E): 4 - 4 = 0$



Escolhemos arcos de custo zero para formar D_0

D_0 não é uma r -arborescência

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

Em outras palavras: há vértices não alcançáveis a partir de r

Passo 2: Contração do Ciclo

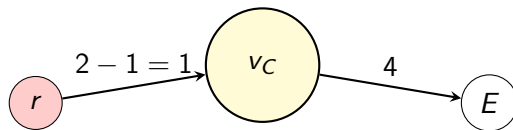
Ciclo C encontrado: $\{A, B, C, D\}$

Contraímos C em supervértice x_C

Em outras palavras: todos os vértices de C viram um único vértice

Custos já foram reduzidos por λ :

$$w_\lambda(r, A) = 2 - 1 = 1$$

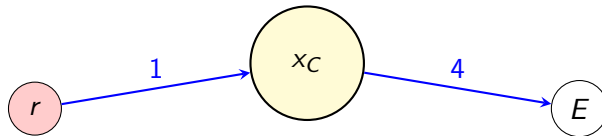


Chamada Recursiva

Resolvemos $\text{chu-liu-edmonds}(D/C \mapsto x_C, w_\lambda/C \mapsto x_C, r)$

Em outras palavras: aplicamos o mesmo algoritmo no digrafo menor

Passo 3: Solução Recursiva no Digrafo Contraído



O algoritmo devolve $T' = \{(r, x_C), (x_C, E)\}$

Em outras palavras: encontramos a r -arborescência ótima no grafo contraído

Próximo passo: expandir T' de volta para obter T no digrafo original

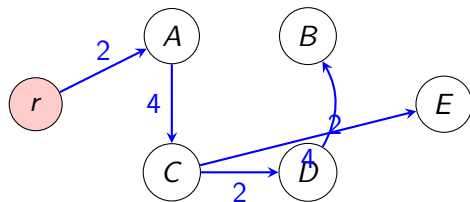
Passo 4: Expansão da Solução

Função $\text{expand}(T')$:

Arco (r, x_C) em T' corresponde a (r, A) em D

Em outras palavras:

- Incluir (r, A) que entra no ciclo
- Incluir todos os arcos do ciclo C **exceto** (B, A)
- Resultado: cada vértice tem grau de entrada 1



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:

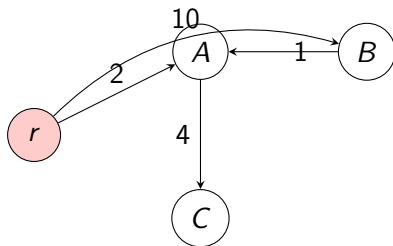
- Trabalha com múltiplos vértices simultaneamente
- Usa componentes fortemente conexas
- Redução sistemática de custos

Complexidade:

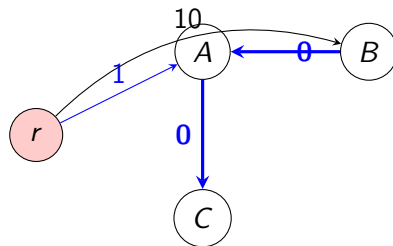
- Fase I: $O(nm)$
- Fase II v1 (lista): $O(n^2)$
- Fase II v2 (heap): $O(n \log n)$

Fase I: Redução de Custos

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



Original



Após Redução

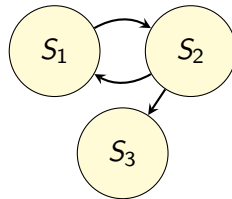
Arcos com custo **zero** formam o digrafo D_0

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 Otimalidade

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

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

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)

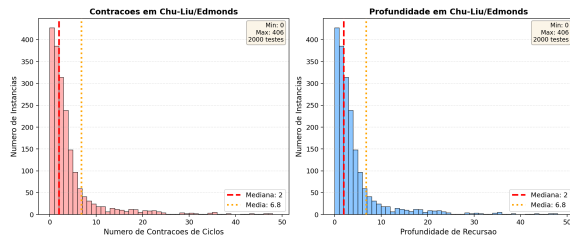
Características Estruturais

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)



Motivação Didática

Desafio

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

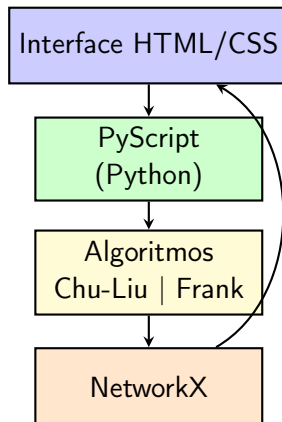
Solução Proposta:

- 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**

 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







Interface: Desenho de Grafos


ArboGraph

Home

Chu-Liu/Edmonds

Andras Frank (V1)

Andras Frank (V2)

Desenhe um grafo

Nossa tese

Dúvidas ?

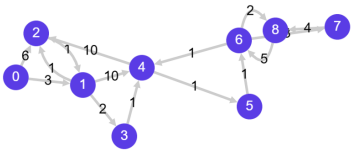
Quer aprender mais sobre esses algoritmos, leia nossa tese :)

Link

Desenhe seu grafo

1. Desenhe um grafo, [carregue](#) um exemplo ou [importe](#) um grafo já existente.

Grafo Original



Funcionalidades:

- Adicionar vértices e arestas
- Definir pesos

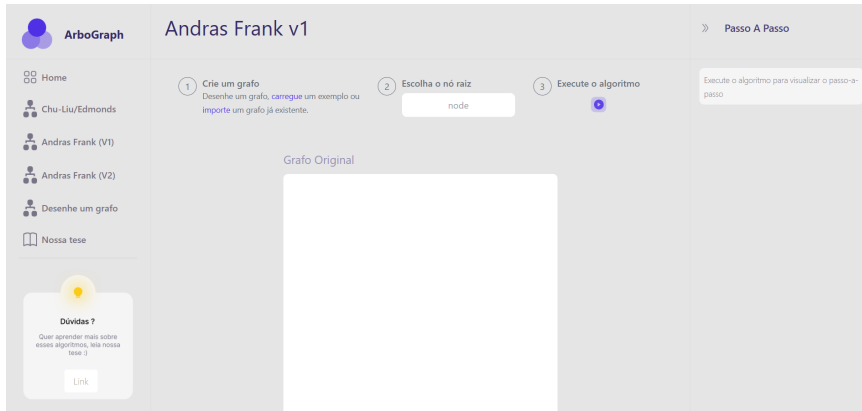
Interface: Chu-Liu-Edmonds

The screenshot shows the ArboGraph web application interface for the Chu-Liu / Edmonds algorithm. The interface is divided into three main sections:

- Left Sidebar:** Contains navigation links: Home, Chu-Liu/Edmonds, Andras Frank (V1), Andras Frank (V2), Desenhe um grafo, and Nossa tese. There is also a "Dúvidas?" (Doubts?) section with a link to learn more about the algorithms.
- Top Section:** Displays the title "Chu-Liu / Edmonds" and a "Passo A Passo" (Step by Step) button.
- Main Content Area:**
 - Step 1: Crie um grafo** (Create a graph): Includes instructions to draw a graph, load an example, or import an existing graph.
 - Step 2: Escolha o nó raiz** (Choose the root node): A dropdown menu showing "0".
 - Step 3: Execute o algoritmo** (Execute the algorithm): A button to execute the algorithm.
 - Grafo Original:** A visual representation of the original graph with 9 nodes (0-8) and weighted edges. The edges and their weights are: (0,1):3, (0,2):6, (1,2):1, (1,3):2, (1,4):10, (2,4):1, (3,4):1, (4,5):1, (4,6):1, (5,6):1, (6,7):8, (6,8):2, (7,8):4, and (8,6):5.
- Right Panel:** A box with the instruction "Execute o algoritmo para visualizar o passo-a-passo" (Execute the algorithm to visualize the step-by-step).

- 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)

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

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: $58\times$ (mediana), $61\times$ (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>