

Lorena Silva Sampaio, Samira Haddad

**Análise e Implementação de Algoritmos de Busca  
de uma  $r$ -Arborescência Inversa de Custo Mínimo  
em Grafos Dirigidos com Aplicação Didática  
Interativa**

Brasil

2025

Lorena Silva Sampaio, Samira Haddad

**Análise e Implementação de Algoritmos de Busca de uma  
r-Arborescência Inversa de Custo Mínimo em Grafos  
Dirigidos com Aplicação Didática Interativa**

Dissertação apresentada à Universidade Federal do ABC como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal do ABC

Orientador: Prof. Dr. Mário Leston

Brasil

2025

*Dedicatória (opcional).*

# Agradecimientos

Agradecimientos (opcional).



# Resumo

Este trabalho apresenta uma análise e implementação de algoritmos de busca de uma  $r$ -arborescência inversa de custo mínimo em grafos dirigidos com aplicação didática interativa.

**Palavras-chave:** Grafos. Arborescência. Algoritmos. Visualização.

# Abstract

This work presents an analysis and implementation of algorithms for finding a minimum cost inverse  $r$ -arborescence in directed graphs with interactive didactic application.

**Keywords:** Graphs. Arborescence. Algorithms. Visualization.

# Lista de ilustrações

Figura 1 – Digrafo original com custos dos arcos e raiz $r$ . . . . .	9
Figura 2 – Execução do algoritmo de András Frank: elevação de potenciais, contração de ciclo e extração da arborescência ótima. . . . .	11

# Sumário

<b>1</b>	<b>ALGORITMO GULOSO–DUAL DE ANDRÁS FRANK</b>	<b>9</b>
<b>1.1</b>	<b>Formulação primal–dual e cortes laminares</b>	<b>10</b>
<b>1.2</b>	<b>Descrição detalhada do algoritmo</b>	<b>10</b>
<b>1.3</b>	<b>Implementação em Python</b>	<b>10</b>
<b>1.4</b>	<b>Exemplo de execução do algoritmo</b>	<b>11</b>
<b>1.5</b>	<b>Complexidade e comparação com Chu–Liu/Edmonds</b>	<b>11</b>
<b>1.6</b>	<b>Notas finais</b>	<b>11</b>
	<b>REFERÊNCIAS</b>	<b>12</b>
	<b>ANEXOS</b>	<b>13</b>
	<b>ANEXO A – ANEXO A</b>	<b>14</b>

# 1 Algoritmo Guloso–Dual de András Frank

Assim como o algoritmo de Chu–Liu/Edmonds, o método de András Frank resolve o problema da  $r$ -arborescência de custo mínimo em digrafos ponderados. A diferença está na abordagem: Frank utiliza uma estrutura gulosa–dual.

O algoritmo de Frank é dividido em duas fases, conforme a descrição abaixo:

## Algoritmo de András Frank

### Fase I — Elevação de potenciais (dual):

- Para cada vértice  $v \neq r$ , eleva-se o potencial  $y(v)$  até que exista ao menos um arco de custo reduzido zero entrando em  $v$ .
- Sempre que um ciclo de arcos de custo zero surgir, contrai-se esse ciclo em um supervértice, ajustando os custos conforme a regra dual.
- O resultado é um subgrafo  $D_0$  onde cada vértice não-raiz tem ao menos uma entrada de custo zero.

### Fase II — Extração primal:

- Seleciona-se exatamente uma entrada de custo zero para cada vértice  $v \neq r$ .
- Se as escolhas formarem um ciclo, contrai-se e repete o processo no grafo menor; ao final, reexpande cada contração removendo uma aresta interna para restaurar a estrutura de arborescência.
- O resultado é uma  $r$ -arborescência ótima, composta apenas por arcos apertados.

Vamos ilustrar o funcionamento das fase 1 e 2 com um exemplo.

Considere o digrafo abaixo, onde os custos dos arcos são indicados e a raiz é  $r$ :

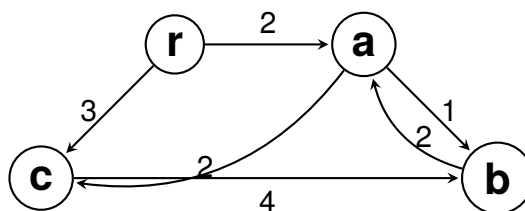


Figura 1 – Digrafo original com custos dos arcos e raiz  $r$ .

Suponha que elevamos o potencial  $y(v)$  de um vértice  $v$  até que o arco  $(u, v)$  tenha

custo reduzido zero:  $c'(u, v) = c(u, v) - y(v) = 0$ . Se todos os arcos que entram em um conjunto de vértices  $C$  ficarem com custo zero, e  $C$  formar um ciclo, contraímos  $C$  em um supervértice  $x_C$  e continuamos o processo no grafo menor.

## 1.1 Formulação primal–dual e cortes laminares

Assim como no capítulo anterior, o problema pode ser formulado como um programa linear primal (seleção de arcos) e seu dual (elevação de potenciais por cortes). A família de cortes ativos pode ser tomada laminar, guiando as contrações e reexpansões.

### Formulação primal–dual do problema de arborescência mínima

**Primal:** Minimizar o custo total dos arcos escolhidos, garantindo conectividade e grau de entrada 1 para cada vértice não-raiz.

**Dual:** Elevar potenciais  $y(X)$  para cortes  $X$  de modo que os custos reduzidos  $c'(u, v) = c(u, v) - \sum_{X: v \in X, u \notin X} y(X)$  permaneçam não negativos.

## 1.2 Descrição detalhada do algoritmo

O algoritmo segue os seguintes passos, espelhando a estrutura do capítulo do Chu–Liu/Edmonds:

### Passos do algoritmo de András Frank

**Passo 1:** Para cada vértice  $v \neq r$ , eleve o potencial  $y(v)$  até que exista ao menos um arco de custo reduzido zero entrando em  $v$ .

**Passo 2:** Sempre que um ciclo de arcos de custo zero surgir, contraia o ciclo em um supervértice, ajustando os custos conforme a regra dual.

**Passo 3:** Repita os passos anteriores até que, no grafo corrente, todo vértice não-raiz tenha ao menos uma entrada de custo zero.

**Passo 4:** Selecione exatamente uma entrada de custo zero para cada vértice não-raiz. Se as escolhas formarem um ciclo, contraia e repita no grafo menor.

**Passo 5:** Ao final, reexpanda cada contração removendo uma aresta interna para restaurar a estrutura de arborescência.

## 1.3 Implementação em Python

Assim como no capítulo anterior, a implementação é modular e didática, com funções auxiliares para cada etapa do algoritmo. Utilizamos a biblioteca NetworkX para manipulação de digrafos e mantemos a clareza e paralelismo com a teoria.

## Funções auxiliares da implementação

**get\_arcs\_entering\_X(D, X):** retorna as arestas que entram em  $X$ .

**get\_minimum\_weight\_cut(arcs):** retorna o peso mínimo entre as arestas que entram em  $X$ .

**update\_weights\_in\_X(D, arcs, min\_weight, A\_zero, D\_zero):** atualiza os pesos das arestas que entram em  $X$ , subtraindo o valor mínimo e registrando as que atingem peso zero.

**has\_arborescence(D, r0):** verifica se o digrafo possui uma arborescência com raiz  $r_0$ .

## 1.4 Exemplo de execução do algoritmo

Considere o digrafo da Figura 1. A Fase I eleva os potenciais até que cada vértice não-raiz tenha ao menos uma entrada de custo zero. Se um ciclo de arcos de custo zero surgir, ele é contraído. A Fase II seleciona uma entrada de custo zero para cada vértice, tratando ciclos por contração e reexpansão, até obter a  $r$ -arborescência ótima.

Figura 2 – Execução do algoritmo de András Frank: elevação de potenciais, contração de ciclo e extração da arborescência ótima.

## 1.5 Complexidade e comparação com Chu–Liu/Edmonds

O algoritmo de Frank tem complexidade  $O(mn)$ , podendo ser otimizado para  $O(m \log n)$  com uso de heaps. A estrutura modular permite comparação direta com o algoritmo de Chu–Liu/Edmonds, tanto em termos de desempenho quanto de clareza conceitual.

## 1.6 Notas finais

Assim como no capítulo anterior, a implementação e análise do algoritmo de András Frank reforçam a importância da abordagem primal–dual e da estrutura laminar de cortes. A comparação sistemática entre os métodos será apresentada na próxima seção, destacando semelhanças, diferenças e aplicações práticas.

## Referências

# Anexos

# ANEXO A – Anexo A

Conteúdo do anexo A.