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

Orientador: Mário Leston
Discentes: Lorena Silva Sampaio, Samira Haddad

10 de Dezembro de 2024

1 Introdução

O problema da *r-arborescência de custo mínimo* consiste em, dado um grafo dirigido com custos nas arestas, encontrar um subgrafo orientado enraizado em um vértice r que conecte r a todos os demais vértices por caminhos direcionados, minimizando a soma dos custos das arestas selecionadas.

Ao longo das últimas décadas, consolidaram-se duas famílias clássicas de métodos: (i) o algoritmo de Chu–Liu/Edmonds, que opera por normalização dos custos das arestas de entrada, seleção sistemática de arcos de custo zero e contração de ciclos até obter um grafo reduzido, seguido de reexpansão para reconstrução da solução [1, 2]; e (ii) a abordagem dual em duas fases de András Frank, fundamentada em cortes dirigidos, na qual se maximiza uma função de cortes c-viável para induzir 0-arestas e, em seguida, extrai-se a arborescência apenas a partir dessas arestas [3]. Embora assentados em princípios distintos — contração de ciclos no plano primal versus empacotamento/dualidade por cortes —, ambos os paradigmas produzem soluções ótimas e tornam explícitas as relações estruturais em grafos dirigidos e a noção de r-arborescência.

Neste trabalho, analisamos e implementamos, em Python, duas abordagens clássicas para o problema de *r-arborescência (inversa) de custo mínimo*: (i) o algoritmo de Chu–Liu/Edmonds, estruturado pela normalização dos pesos das arestas de entrada, seleção de 0-arestas, detecção/contração de ciclos e posterior reexpansão; e (ii) o método dual de András Frank em duas fases, que maximiza uma função de cortes c-viável para induzir 0-arestas (fase 1) e extrai a arborescência apenas a partir dessas arestas (fase 2). A dissertação articula os fundamentos teóricos com suas traduções algorítmicas em Python, discutindo detalhadamente as decisões de projeto e as etapas de implementação.

Adicionalmente, desenvolvemos uma aplicação web com fins didáticos, utilizando o framework PyScript, e as bibliotecas NetworkX e Matplotlib, que permitem construir grafos direcionados interativamente, escolher o vértice-raiz, executar o algoritmo de Chu–Liu/Edmonds e acompanhar, passo a passo, a evolução do grafo e o registro detalhado da execução (log). A interface inclui operações de adicionar arestas com pesos, carregar um

grafo de teste e exportar a instância em formato JSON, facilitando a experimentação por estudantes e educadores.

1.1 Justificativa

A escolha do problema de *r-arborescência de custo mínimo* se justifica pela sua relevância em diversas aplicações práticas, como otimização de redes de comunicação, planejamento de rotas e análise de estruturas hierárquicas. Além disso, a comparação entre diferentes abordagens algorítmicas para a resolução deste problema permite uma compreensão mais profunda das técnicas de otimização em grafos.

1.2 Objetivos

O objetivo principal deste trabalho é analisar, implementar e comparar duas abordagens clássicas para o problema de *r-arborescência de custo mínimo* em grafos dirigidos.

Além disso, buscamos desenvolver uma aplicação web interativa que facilite o entendimento e a experimentação com o algoritmo de Chu–Liu/Edmonds e o método de András Frank, tornando-o acessível para estudantes e educadores.

1.3 Estrutura do Trabalho

Resumidamente, o trabalho abrange as seguintes frentes:

1. **Análise teórica**: consolidação dos conceitos de dígrafos e arborescências, compondo as formulações primal (normalização de custos e contração/reexpansão de ciclos no algoritmo de Chu–Liu/Edmonds) e dual (cortes dirigidos e função c-viável no método de András Frank), destacando resultados e intuições estruturais.

2. **Implementação computacional**: implementação em Python das rotinas de normalização dos custos de entrada, construção de F^*, detecção e contração de ciclos e reconstrução da solução (Chu–Liu/Edmonds), bem como das duas fases do método de András Frank; além de uma suíte de testes automatizados em larga escala sobre instâncias aleatórias com até centenas de vértices, verificando a coincidência dos custos entre os métodos e registrando resultados em CSV e log.

3. **Aplicação pedagógica**: desenvolvimento de uma aplicação web interativa (PyScript + NetworkX + Matplotlib) que permite montar instâncias, escolher o vértice-raiz e acompanhar, passo a passo, a execução do algoritmo com visualização do grafo e dos pesos das arestas, log textual e importação/exportação em JSON para facilitar a reprodução de experimentos.

Como validação empírica, realizamos testes de volume com milhares de instâncias geradas aleatoriamente, registrando resultados em arquivos CSV e de log; os custos obtidos pelo Chu–Liu/Edmonds e pelas duas variantes de András Frank coincidem, corroborando a correção das implementações. Deste modo, o trabalho entrega implementações verificadas de Chu–Liu/Edmonds e András Frank, um visualizador web interativo e testes de volume que confirmam a equivalência de custos, úteis ao estudo e ao ensino de arborescências.

2 Definições Preliminares

Neste capítulo, reunimos as noções básicas de teoria dos grafos dirigidos utilizadas ao longo do texto. O objetivo é fixar notações e conceitos (conjuntos, relações, funções, dígrafos, propriedade em dígrafos, dígrafos ponderados, ramificações geradoras, arborescências, funções de custo, dualidade, problemas duais e algoritmos), até chegar à formulação do problema da r-arborescência inversa de custo mínimo e adiamos descrições algorítmicas para capítulos posteriores.

2.1 Conjuntos

Este trabalho depende profundamente da teoria dos conjuntos. Basicamente podemos dizer que todos os objetos matemáticos que iremos utilizar nessa dissertação se reduzem a conjuntos e operações entre eles. Um conjunto é uma agregação de objetos distintos com características bem definidas, chamados elementos ou membros do conjunto. Os conjuntos são geralmente representados por letras maiúsculas (por exemplo, A, B, C) e seus elementos são listados entre chaves (por exemplo, $A = \{1, 2, 3\}$). Dois conjuntos são iguais se contêm exatamente os mesmos elementos.

Podemos ter conjuntos de qualquer tipo de objeto, incluindo números, letras, elementos da natureza, etc. Para motivar as definições ao longo do texto, adotaremos um exemplo-mestre com organismos: árvores, plantas e fungos, usando pertinência e inclusão para guiar a intuição.

2.1.1 Pertinência e inclusão

Começamos pela **noção de pertinência**: dizemos que um elemento x pertence a um conjunto X quando $x \in X$ e não pertence quando $x \notin X$.

Para ancorar ideias, fixemos um universo U de organismos e definamos três conjuntos: $P = \{\text{todas as plantas}\}$, $T = \{\text{todas as árvores}\}$ e $F = \{\text{todos os fungos}\}$. Se x é um carvalho, então $x \in T$ e, como toda árvore é uma planta, $x \in P$. Já se y é um cogumelo, então $y \in F$ e, na taxonomia moderna, $y \notin T$ e $y \notin P$. Agora, considere $A = \{\text{árvores com folhas verdes}\}$; por definição, $A \subseteq T$ e a pertinência fica clara: $x \in A$ se, e somente se, x é árvore e tem folhas verdes.

Em seguida, vem a **relação de inclusão** entre conjuntos: escrevemos $X \subseteq Y$ quando todo elemento de X também pertence a Y (e $X \subset Y$ quando, além disso, $X \neq Y$). No nosso exemplo, $A \subseteq T \subset P$ e $T \cap F = \varnothing$ (árvores e fungos não se sobrepõem).

2.1.2 Operações entre conjuntos

Com essas noções de pertinência e inclusão, apresentamos as operações básicas entre conjuntos, que usaremos ao longo do texto (mantendo o exemplo com P, T, F, A).

Outras operações comuns entre conjuntos incluem:

- **União** ($A \cup B$): o conjunto de todos os elementos que pertencem a A, a B, ou ambos.

  Exemplo: $T \cup F$ é o conjunto de todos os organismos que são árvores ou fungos (ou ambos, se existissem tais organismos).

- **União disjunta** ($A \uplus B$): o conjunto de todos os elementos que pertencem a A ou a B, mas não a ambos; é igual a $A \cup B$ quando A e B são disjuntos.

  Exemplo: $T \uplus F$ é o conjunto de todos os organismos que são árvores ou fungos, mas não ambos (o que é trivialmente igual a $T \cup F$ pois T e F são disjuntos).

- **Interseção** ($A \cap B$): o conjunto de todos os elementos que pertencem tanto a A quanto a B.

  Exemplo: $T \cap P = T$, pois todas as árvores são plantas.

- **Diferença** ($A \setminus B$): o conjunto de todos os elementos que pertencem a A mas não a B.

  Exemplo: $T \setminus A$ é o conjunto de todas as árvores que não têm folhas verdes.

- **Complemento** de X em um universo fixo U: $X^c := U \setminus X$ (também chamado de *complemento absoluto*); o *complemento relativo* de X em Y é $Y \setminus X$.

  Exemplo: $T^c = U \setminus T$ é o conjunto de todos os organismos que não são árvores. Ou seja, T^c inclui plantas que não são árvores, fungos e quaisquer outros organismos no universo U.

- **Diferença simétrica** ($A \Delta B$): $(A \setminus B) \cup (B \setminus A)$; é igual a $A \cup B$ quando A e B são disjuntos.

  Exemplo: $P \Delta F$ é o conjunto de todos os organismos que são plantas ou fungos, mas não ambos (o que é trivialmente igual a $P \cup F$ pois P e F são disjuntos).

- **Produto cartesiano** ($A \times B$): o conjunto de pares ordenados (a, b) com $a \in A$ e $b \in B$.

  Exemplo: suponha $T = \{t_1, t_2\}$ e $F = \{f_1, f_2\}$. Então $T \times F = \{(t_1, f_1), (t_1, f_2), (t_2, f_1), (t_2, f_2)\}$.

- **Conjunto das partes** (2^U): a família de todos os subconjuntos de U (inclui $\varnothing$ e o próprio U).

  Exemplo: se $U = \{x, y\}$, então $2^U = \{\varnothing, \{x\}, \{y\}, \{x, y\}\}$. Logo, $|2^U| = 4 = 2^{|U|}$.

Identidades úteis. Usaremos livremente as propriedades clássicas de conjuntos — comutatividade e associatividade de $\cup$ e $\cap$, distributividade e as **leis de De Morgan** — sem prova. Quando for relevante, explicitaremos a identidade no ponto de uso. Por exemplo, no nosso universo U, $(P \cup F)^c = P^c \cap F^c$.

Referências

[1] Y. J. Chu e T. H. Liu. "On the Shortest Arborescence of a Directed Graph". Em: *Scientia Sinica* 14 (1965), pp. 1396–1400.

[2] J. Edmonds. "Optimum Branchings". Em: *Journal of Research of the National Bureau of Standards* 71B (1967), pp. 233–240.

[3] A. Frank e G. Hajdu. "A Simple Algorithm and Min–Max Formula for the Inverse Arborescence Problem". Em: *Algorithms* 7.4 (2014), pp. 637–647. DOI: 10.3390/a7040637.