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

21 de setembro de 2025

1 Introdução

Encontrar uma *r-arborescência inversa de custo mínimo* em grafos dirigidos é um problema estudado em ciência da computação desde os anos 1960, com formulações fundamentais apresentadas por Jack Edmonds em 1967 [**edmonds1967optimum**].

Essa busca dialoga com um princípio formulado na Idade Média por Guilherme de Ockham: a navalha de Occam (princípio da parcimônia), uma heurística filosófica segundo a qual, entre explicações concorrentes para um fenômeno, devemos preferir a mais simples ou a que faz menos suposições.

Podemos pensar na navalha de Occam como critério de escolha entre explicações por meio de uma *teia explicativa mínima*: uma estrutura que conecta fatos ou observações com o mínimo de relações explicativas necessárias. Quando tais relações envolvem dependência ou causalidade, podemos representá-las pictograficamente como setas direcionadas entre os fatos (as hipóteses aparecem como rótulos dessas setas). Para refinar o modelo, associamos um custo a cada relação (por exemplo, o esforço para validar a relação ou a complexidade da explicação).

Encontrar a *teia explicativa mínima* equivale, nessa metáfora, a encontrar uma *r-arborescência de custo mínimo*: fixamos um vértice raiz r (a explicação inicial) e escolhemos um conjunto mínimo de relações explicativas de modo que todos os fatos tenham um caminho dirigido que leve a r, minimizando o custo total das arestas.

A *r-arborescência* (também chamada *out-arborescência*) orienta as arestas para fora de r: cada vértice $v \neq r$ tem exatamente uma aresta de entrada, e há um caminho dirigido único de r até v. Já a *r-arborescência inversa* (*in-arborescência*) orienta as arestas em direção a r: cada $v \neq r$ tem exatamente uma aresta de saída, e de cada vértice parte um caminho dirigido único até r [**edmonds1967optimum**, **frank2014**].

Com essa distinção em mente, este trabalho concentra-se na variante inversa. Formalmente, dado um grafo dirigido $G = (V, E)$ com custos $c : E \to \mathbb{R}^+$ nas arestas e um vértice raiz $r \in V$, procura-se uma *r-arborescência inversa* — isto é, uma árvore direcionada que atinja todos os vértices por caminhos dirigidos até r — que minimize o custo total das arestas selecionadas (cf. [**edmonds1967optimum**, **frank2014**]).

Nosso interesse, porém, não é apenas encontrar a arborescência mínima: o percurso até ela também importa, pois revela propriedades estruturais dos dígrafos e ilumina técnicas distintas de otimização. Por isso, investigamos duas rotas clássicas e complementares: (i) o algoritmo de Chu–Liu/Edmonds, que opera por normalização dos custos das arestas de entrada, seleção sistemática de arestas de custo zero e contração de ciclos até obter um grafo reduzido, seguida pela reexpansão para reconstrução da solução [**chu1965**, **edmonds1967optimum**]; 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 arestas de custo zero e, em seguida, extrai-se a arborescência apenas a partir dessas arestas [**frank2014**]. 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 a variedade de abordagens matemáticas que podem ser empregadas para resolver o mesmo problema.

Assim sendo, analisamos e implementamos, em Python, essas duas abordagens, e apresentaremos detalhes das implementações, desafios enfrentados e soluções adotadas. 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.

Adicionalmente, desenvolvemos uma aplicação web com fins didáticos, utilizando o framework PyScript e as bibliotecas NetworkX e Matplotlib, que permitem construir grafos dirigidos 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 busca por uma *r-arborescência inversa de custo mínimo* em grafos dirigidos é um problema clássico com aplicações em diversas áreas, como redes de comunicação, planejamento de rotas, análise de dependências e modelagem de processos. Mas, não precisamos dessa justificação prática para nos interessarmos pelo problema: a riqueza estrutural dos dígrafos e a variedade de técnicas algorítmicas disponíveis o tornam um excelente caso de estudo em otimização combinatória.

Do ponto de vista didático, a metáfora da "teia explicativa mínima" torna concreto o porquê de estudarmos arborescências enraizadas: ela mapeia perguntas sobre explicação, alcance e economia de recursos para estruturas dirigidas, servindo de fio condutor nas implementações e nos experimentos que apresentamos.

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 oferecendo 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. **Fundamentação teórica**: revisão da literatura sobre arborescências em grafos dirigidos, incluindo definições, propriedades e resultados relevantes.

2. **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.

3. **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.

4. **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.

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 matemáticas básicas necessárias para compreensão completa do texto.

Fixaremos 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, 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 e elementos da natureza. Para motivar as definições ao longo do texto, usaremos dois exemplos complementares que vamos chamar de exemplos-mestres:

- (i) Considere um universo U composto por três conjuntos: árvores T, plantas P e fungos F — para praticar pertinência, inclusão e operações; e

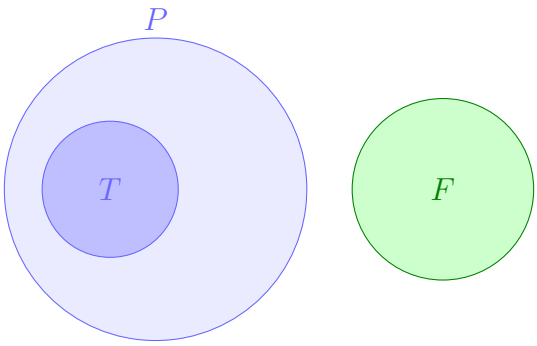


Figura 1: Relações entre os conjuntos de organismos: $T \subseteq P$ (toda árvore é planta) e F é disjunto de P.

- (ii) exemplo inspirado na navalha de Occam, com três famílias: evidências E, hipóteses H e explicações $\mathcal{M} \subseteq 2^H$. Nesse segundo exemplo, privilegiaremos explicações parcimoniosas: entre as que cobrem E, preferimos as minimais por inclusão.

No segundo exemplo, consideremos $E = \{\text{queda de temperatura, céu nublado}\}$ e $H = \{H_A, H_B\}$, em que H_A significa "frente fria" e H_B, "ilha de calor". Tanto $\{H_A\}$ quanto $\{H_A, H_B\}$ explicam E (cobrem ambas as evidências), mas, por parcimônia, preferimos $\{H_A\}$, por ser estritamente menor do que $\{H_A, H_B\}$ ($\{H_A\} \subset \{H_A, H_B\}$). Ao longo do texto, recorreremos às noções de pertinência (por exemplo, $H_A \in H$), de inclusão e às operações usuais sobre conjuntos (união, interseção etc.) para comparar explicações.

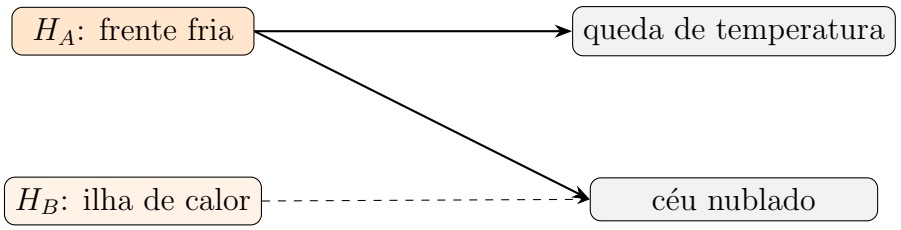


Figura 2: Exemplo inspirado na navalha de Occam: H_A cobre ambas as evidências (E), enquanto H_B seria redundante; prefere-se a explicação menor.

2.1.1 Subconjuntos

Dizemos que A é um **subconjunto** de B, denotado $A \subseteq B$, quando todo elemento de A também pertence a B. Se, além disso, $A \neq B$, escrevemos $A \subset B$ e chamamos A de **subconjunto próprio** de B. Ex.: $\{1, 2\} \subseteq \{1, 2, 3\}$ e $\{1, 2\} \subset \{1, 2, 3\}$.Por convenção, o conjunto vazio $\varnothing$ é subconjunto de qualquer conjunto X (isto é, $\varnothing \subseteq X$), e todo conjunto é subconjunto de si mesmo ($X \subseteq X$).

No primeiro exemplo-mestre, sejam P o conjunto de plantas, T o de árvores e F o de fungos; então $T \subseteq P$ (toda árvore é planta), ao passo que $F \nsubseteq P$.

No segundo, seja $H = \{H_A, H_B\}$ e $E = \{\text{queda de temperatura, céu nublado}\}$, vale $\{H_A\} \subset \{H_A, H_B\} \subseteq H$; ambas as opções explicam E, mas, por parcimônia, preferimos $\{H_A\}$.

2.1.2 Pertinência e inclusão

Pertinência e inclusão são os conceitos mais fundamentais da teoria dos conjuntos.

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

Seja o nosso universo U de organismos: $P = \{\text{todas as plantas}\}$, $T = \{\text{todas as árvores}\}$ 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 $A = \{\text{árvores com folhas verdes}\}$; a pertinência fica clara: $x \in A$ se, e somente se, x é árvore e tem folhas verdes.

Continuando, vem a **relação de inclusão** entre conjuntos denotada por $\subseteq$: 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).

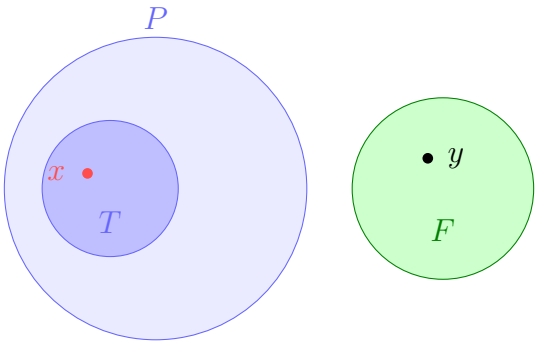


Figura 3: Pertinência: $x \in T \subseteq P$ (ponto vermelho dentro de T) e $y \in F$ (ponto preto); logo $y \notin P$ e $y \notin T$.

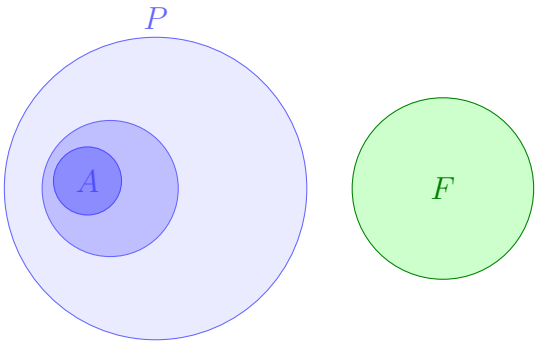


Figura 4: Inclusão: $A \subseteq T \subset P$ (círculos aninhados) e $T \cap F = \varnothing$ (círculos disjuntos).

2.1.3 Operações entre conjuntos

Com essas definições de pertinência, inclusão e subconjuntos, 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).

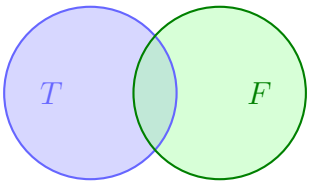


  Figura 5: União: a área colorida representa $T \cup F$; a sobreposição evidencia a intersecção $T \cap F$ (hipotética).

- **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).

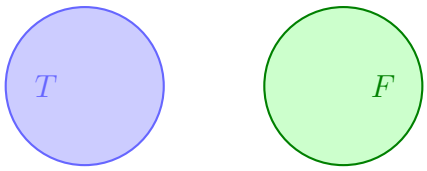


Figura 6: União disjunta: como $T \cap F = \varnothing$, tem-se $T \uplus F = T \cup F$.

- **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.

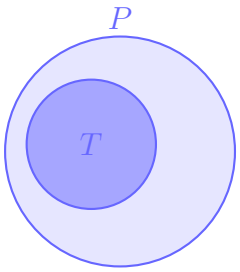


Figura 7: Interseção: como $T \subseteq P$, $T \cap P = T$ (a região escura é T).

- **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.

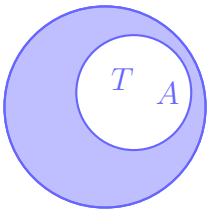


Figura 8: Diferença: região azul representa $T \setminus A$ (árvores que não têm folhas verdes).

- **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.

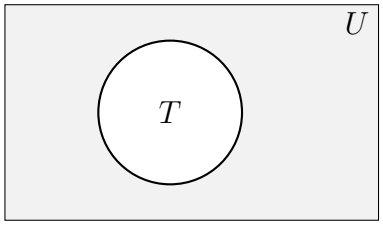


Figura 9: Complemento: a área cinza representa $T^c = U \setminus T$.

- **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).

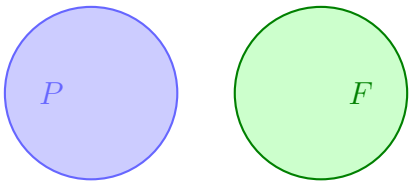


Figura 10: Diferença simétrica: como $P \cap F = \varnothing$, temos $P \,\Delta\, F = P \cup F$.

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

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

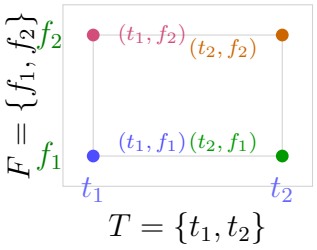


Figura 11: Produto cartesiano: pontos representam os pares de $T \times F$ para $T = \{t_1, t_2\}$ e $F = \{f_1, f_2\}$.

- **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, $|2^U| = 4 = 2^{|U|}$.

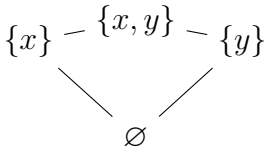


Figura 12: Conjunto das partes: diagrama de Hasse de 2^U para $U = \{x, y\}$.

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, $(P \cup F)^c = P^c \cap F^c$.

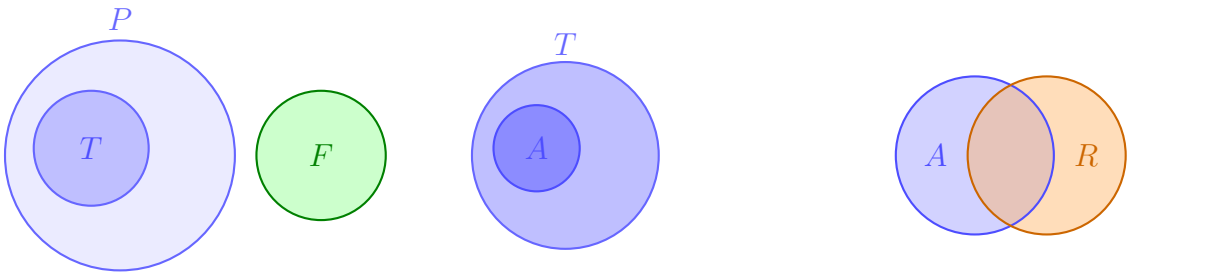
2.1.4 Coleção

Entre os objetos que podem pertencer a um conjunto, estão também eles mesmos, outros conjuntos. Chamaremos tais conjuntos de **coleções** (ou **famílias**) de conjuntos. Por exemplo, $\mathcal{C} = \{P, T, F\}$ é uma coleção formada pelos conjuntos de organismos já definidos: plantas P, árvores T e fungos F. Note que $\mathcal{C}$ é um conjunto como outro qualquer; seus elementos são, cada um, um conjunto.

Coleções são úteis para agrupar subconjuntos relacionados de um mesmo universo. Por exemplo, considere $\mathcal{D} = \{A, B\}$, onde $A = \{\text{árvores com folhas verdes}\}$ e $B = \{\text{árvores com folhas vermelhas}\}$. Assim, $\mathcal{D} \subseteq 2^T$ é uma coleção de subconjuntos de T.

Uma coleção $\mathcal{F}$ é dita **laminar** quando, para quaisquer $X, Y \in \mathcal{F}$, vale que $X \subseteq Y$, $Y \subseteq X$ ou $X \cap Y = \varnothing$; isto é, quaisquer dois conjuntos são aninhados (um está contido no outro) ou são disjuntos.

Por exemplo, na coleção $\mathcal{C} = \{P, T, F\}$: P é o conjunto de todas as plantas, T o de todas as árvores (portanto $T \subseteq P$) e F o de todos os fungos (disjunto de plantas e, logo, de árvores). Assim, quaisquer dois conjuntos em $\mathcal{C}$ são aninhados ou disjuntos, e $\mathcal{C}$ é laminar. Na coleção $\mathcal{D} = \{A, T\}$: A é o conjunto de árvores com folhas verdes e T o de todas as árvores; como toda árvore de A é árvore de T, temos $A \subseteq T$ e a coleção é laminar. Já em $\mathcal{E} = \{A, R\}$: R é o conjunto de árvores frutíferas; há árvores que são ao mesmo tempo frutíferas e de folhas verdes (a interseção é não vazia), mas nenhuma das classes contém a outra, então $\mathcal{E}$ não é laminar.



(a) $\mathcal{C} = \{P, T, F\}$ — laminar (b) $\mathcal{D} = \{A, T\}$ — laminar (c) $\mathcal{E} = \{A, R\}$ — não laminar

Figura 13: Laminaridade em coleções: em (a) e (b), quaisquer dois conjuntos são aninhados ou disjuntos; em (c), A e R se interceptam sem inclusão, violando a laminaridade.

Este é um importante conceito que aparecerá no restante do trabalho. A ideia de laminaridade retornará quando tratarmos de cortes dirigidos.

2.1.5 Comparando conjuntos: cardinalidade e maximalidade

Podemos comparar conjuntos através de relações de tamanho (cardinalidade) ou por relações de inclusão. Essas duas formas de comparação são distintas e importantes, especialmente quando lidamos com coleções de conjuntos.

A **cardinalidade** de um conjunto A, denotada por $|A|$, é o número de elementos de A. Para conjuntos finitos, é simplesmente a contagem dos elementos (por exemplo, se $A = \{1, 2, 3\}$, então $|A| = 3$). Para conjuntos infinitos, a cardinalidade pode ser mais complexa, envolvendo conceitos como infinito enumerável e não enumerável. Por exemplo, o conjunto dos números naturais $\mathbb{N}$ é infinito enumerável, enquanto o conjunto dos números reais $\mathbb{R}$ é infinito não enumerável.

Dizemos que $A \in \mathcal{C}$ tem **maior cardinalidade** se $|A| \geq |B|$ para todo $B \in \mathcal{C}$ (podendo haver empates). Esse critério não coincide, em geral, com a comparação por relação de inclusão. Em grafos, por exemplo, distinguem-se conjuntos independentes *maximais* (não ampliáveis) de conjuntos independentes *máximos* (de cardinalidade máxima).

Ao compararmos uma coleção $\mathcal{C}$ de conjuntos utilizando sua relações de inclusão $(\mathcal{C}, \subseteq)$, é imprescindível distinguir **maximal** de **máximo**.

Um conjunto $A \in \mathcal{C}$ é **maximal** se não existe $B \in \mathcal{C}$ tal que $A \subset B$. Em palavras: não dá para ampliar A estritamente dentro da coleção. Podem haver vários elementos maximais, e eles podem ser incomparáveis entre si. Ex.: em $\mathcal{C} = \big\{\{1\}, \{2\}\big\}$, ambos $\{1\}$ e $\{2\}$ são maximais, mas não existe máximo.

Um conjunto $A \in \mathcal{C}$ é **máximo** se $B \subseteq A$ para todo $B \in \mathcal{C}$. Se existe, é único. Ex.: em $\mathcal{C} = \big\{\{1\}, \{2\}, \{1, 2\}\big\}$, o conjunto $\{1, 2\}$ é o máximo.

Um bom exemplo para ilustrar a distinção entre conjuntos maximais e máximos é a coleção $\mathcal{C} = \big\{\{1\}, \{2\}, \{1,2\}, \{3\}\big\}$. Aqui, $\{1,2\}$ é o único conjunto máximo (contém todos os outros), enquanto $\{1\}$, $\{2\}$ e $\{3\}$ são todos maximais (não podem ser ampliados dentro da coleção).

Esses conceitos reaparecerão ao longo do texto, especialmente na diferença entre estruturas **maximais** (saturadas por inclusão) e **máximas/ótimas** (de maior cardinalidade ou menor custo). Para fixar ideias:

- Em muitos problemas, "**maximal**" quer dizer: não dá para ampliar uma escolha sem violar as regras; já "**máximo/ótimo**" quer dizer: entre todas as escolhas válidas, essa é a melhor segundo o critério (por exemplo, menor custo).
- No algoritmo de **Chu–Liu/Edmonds**, começamos com escolhas locais que já não podem ser ampliadas dentro das regras do problema e, a partir delas, chegamos a uma solução de menor custo.
- No método de **András Frank**, primeiro construímos uma estrutura organizada que garante escolhas suficientes; depois, usando apenas relações já ativadas por essa estrutura, extraímos a solução ótima.
- Moral: partimos da ideia de "não dá para aumentar" (maximal) e chegamos a "melhor possível" (máximo/ótimo). Os detalhes técnicos de cada método aparecerão nas seções próprias.

2.2 Relações e Funções

Desde a introdução, vimos a ideia filosófica de explicar como "ligar" fatos a hipóteses da forma mais parcimoniosa possível. Para tornar essa intuição precisa, precisamos de uma linguagem que descreva objetos (conjuntos) e como eles se conectam. É aqui que entram as **relações** e, de modo ainda mais disciplinado, as **funções**: regras que associam a cada elemento de um conjunto exatamente um elemento de outro. Com elas, passamos do discurso qualitativo sobre explicações para uma estrutura matemática que permite medir, comparar e, adiante, otimizar.

Na matemática, uma **relação** entre dois conjuntos A e B é uma maneira de associar elementos de A com elementos de B. Uma **função** é um tipo especial de relação que associa cada elemento de A a exatamente um elemento de B.

Uma **relação** R entre dois conjuntos A e B é um subconjunto do produto cartesiano $A \times B$. Ou seja, $R \subseteq A \times B$. Se $(a, b) \in R$, dizemos que a está relacionado a b pela relação R, denotado aRb.

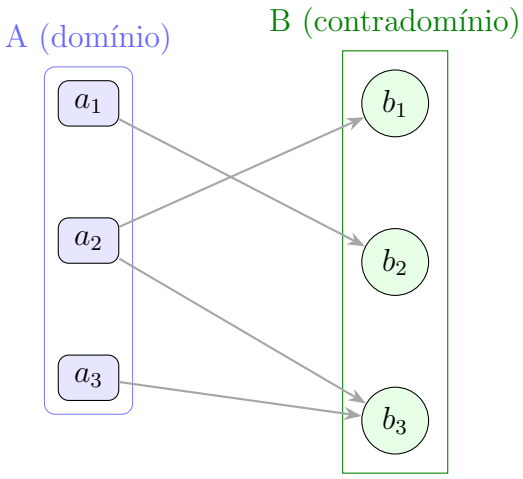


Figura 14: Relação $R \subseteq A \times B$. Cada seta representa um par $(a, b) \in R$ (isto é, $a\,R\,b$). As formas/cores distinguem domínio (A, retângulos azuis) de contradomínio (B, círculos verdes). Note que a_2 se relaciona com b_1 e b_3; logo, este R *não* é uma função. Para que R fosse uma função $f : A \to B$, cada $a \in A$ deveria ter *exatamente uma* seta saindo para algum $b \in B$.

No nosso primeiro exemplo-mestre, considere $P = \{\text{todas as plantas}\}$ e $F = \{\text{todos os fungos}\}$. Definimos a relação R como "é um organismo que compete com". Assim, se uma planta $p \in P$ compete com um fungo $f \in F$, então $(p, f) \in R$.

No nosso segundo exemplo, considere $H = \{\text{hipóteses}\}$ e $E = \{\text{evidências}\}$. Definimos a relação R como "explica". Se uma hipótese $h \in H$ explica uma evidência $e \in E$, então $(h, e) \in R$.

Em teoria dos grafos, uma relação pode representar conexões entre vértices. Por exemplo, em um grafo dirigido, a relação "existe uma aresta de u para v"pode ser representada como um conjunto de pares ordenados (u, v).

Uma **função** f de um conjunto A em um conjunto B é uma relação especial que associa cada elemento de A a exatamente um elemento de B. Denotamos isso como $f : A \to B$. Se $f(a) = b$, dizemos que b é a imagem de a sob f.

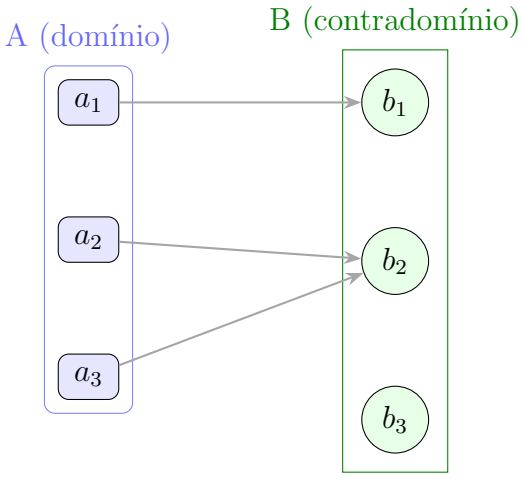


Figura 15: Função $f : A \to B$. Cada elemento de A tem *exatamente uma* seta saindo para um elemento de B (a imagem). Elementos distintos de A podem ter a mesma imagem (muitos-para-um), e nem todo elemento de B precisa ser imagem (aqui, b_3 não é atingido). Compare com a Fig. 14.

No nosso exemplo-mestre, considere $P = \{\text{todas as plantas}\}$ e $\mathbb{N} = \{0, 1, 2, \ldots\}$ (números naturais). Definimos a função $f : P \to \mathbb{N}$ que associa cada planta ao seu número de folhas. Se $p \in P$ é uma árvore com 100 folhas, então $f(p) = 100$.

Em teoria dos grafos, funções podem ser usadas para atribuir pesos ou capacidades às arestas. Por exemplo, se temos um grafo G com arestas $e_1, e_2, \ldots, e_n$, podemos definir uma função $c : E \to \mathbb{R}^+$ que atribui um peso $c(e_i)$ a cada aresta e_i.

Na ciência da computação, relações e funções são usadas para modelar conexões entre dados, estruturas de dados e operações. Por exemplo, em bancos de dados relacionais, tabelas representam relações entre diferentes entidades. Em programação funcional, funções são tratadas como cidadãos de primeira classe, permitindo a criação de funções de ordem superior que podem receber outras funções como argumentos ou retorná-las como resultados.

2.2.1 Conceitos em Funções

Alguns conceitos importantes relacionados a funções incluem:

- **Domínio**: o conjunto A de entrada da função $f : A \to B$.
- **Contradomínio**: o conjunto B de possíveis saídas da função.
- **Imagem**: o conjunto de valores efetivamente atingidos pela função, $f(A) = \{f(a) \mid a \in A\}$.

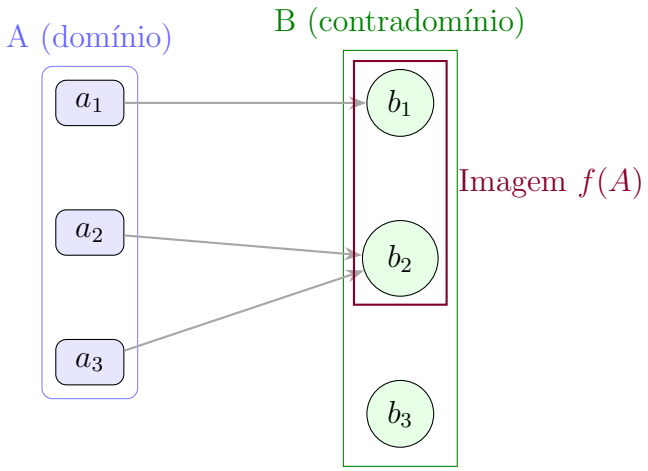


Figura 16: Domínio, contradomínio e imagem: A (retângulos azuis) mapeia via f para B (círculos verdes). A imagem $f(A)$ é o subconjunto de B efetivamente atingido (aqui, $\{b_1, b_2\}$).

- **Injetora**: uma função f é injetora se $f(a_1) = f(a_2)$ implica $a_1 = a_2$; ou seja, elementos distintos do domínio têm imagens distintas.
- **Sobrejetora**: uma função f é sobrejetora se para todo $b \in B$, existe $a \in A$ tal que $f(a) = b$; ou seja, a imagem é igual ao contradomínio.
- **Bijetora**: uma função que é tanto injetora quanto sobrejetora; estabelece uma correspondência um-para-um entre os elementos de A e B.

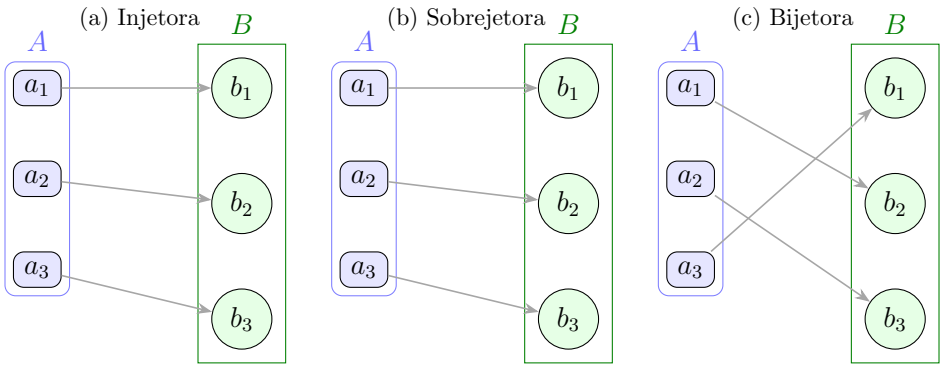


Figura 17: Funções especiais: (a) Injetora — elementos distintos em A têm imagens distintas em B; (b) Sobrejetora — todo elemento de B é imagem; (c) Bijetora — um-para-um e sobre B.

2.2.2 Funções de agregação e somatórios

Além de relacionar elementos de conjuntos, muitas operações familiares em matemática, envolvem *funções* que recebem coleções de números (ou funções) e devolvem um número.

Uma **função de agregação** é uma função que recebe um conjunto (ou sequência) de valores e retorna um único valor que representa algum aspecto agregado desses valores. Exemplos comuns incluem:

- **Média**: A média aritmética de um conjunto de números $x_1, x_2, \ldots, x_n$ é dada por $\frac{1}{n}\sum_{i=1}^{n} x_i$.
- **Máximo e Mínimo**: A função máximo retorna o maior valor em um conjunto, enquanto a função mínimo retorna o menor valor.
- **Produto**: O produto de um conjunto de números $x_1, x_2, \ldots, x_n$ é dado por $\prod_{i=1}^{n} x_i$.
- **Contagem**: A função contagem retorna o número de elementos em um conjunto.

O **somatório**, por exemplo, é uma função de agregação linear que mapeia uma sequência $(x_1, \ldots, x_n)$ em sua soma:

$$\sum_{i=1}^{n} x_i.$$

Esse conceito é especialmente útil em otimização e em análise combinatória: somatórios aparecem o tempo todo e serão explorados ao longo deste trabalho.

Exemplos com grafos. As sessões seguintes explorarão em maiores detalhes grafos e dígrafos, mas agora, consideremos a ideia básica: um **grafo** é um conjunto de pontos (*vértices*) ligados por linhas (*arestas*). No caso *não dirigido*, as linhas não têm seta; no caso *dirigido*, cada linha tem um sentido e é chamada de *arco*.

A noção de somatória aparecerá naturalmente quando lidamos com propriedades dos grafos. Por exemplo:

Seja um grafo não dirigido $G = (V, E)$. O **grau** de um vértice $v \in V$, escrito $\deg(v)$, é quantas arestas tocam em v. A soma dos graus de todos os vértices conta cada aresta *duas vezes* (uma por extremidade), portanto:

$$\sum_{v \in V} \deg(v) = 2\,|E|.$$

Em grafos dirigidos, distinguimos $\deg^-(v)$ (quantos arcos *chegam* em v) e $\deg^+(v)$ (quantos arcos *saem* de v). Cada arco contribui com 1 para um grau de saída e 1 para um grau de entrada, logo:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|.$$

Agora suponha que cada aresta/arco $e \in E$ tenha um *peso* (ou *custo*) $c(e) \geq 0$. O **custo total** de um subconjunto $F \subseteq E$ é simplesmente a soma dos pesos das arestas escolhidas:

$$C(F) = \sum_{e \in F} c(e).$$

De maneira análoga, se $X \subseteq V$ é um conjunto de vértices, a **capacidade** dos arcos que *saem* de X é a soma dos pesos dessas setas. Usaremos mais adiante a notação $\delta^+(X)$ para o conjunto de arcos que saem de X (ver a seção de dígrafos); com essa notação,

$$\mathrm{cap}(X) = \sum_{e \in \delta^+(X)} c(e).$$

Esses exemplos mostram como somatórios capturam propriedades estruturais do grafo por meio de funções simples de agregação.

2.2.3 Funções Especiais

Função de custo

Uma **função de custo** é uma função $c : A \to \mathbb{R}^+$ que atribui um valor numérico não negativo (custo) a cada elemento de um conjunto A. Essas funções são amplamente utilizadas em otimização, economia e teoria dos grafos para modelar despesas, penalidades ou recursos associados a escolhas ou ações.

Exemplo: Considere um conjunto de tarefas $T = \{t_1, t_2, t_3\}$. Uma função de custo $c : T \to \mathbb{R}^+$ pode ser definida como:

$$c(t_1) = 5, \quad c(t_2) = 10, \quad c(t_3) = 3.$$

Aqui, $c(t_i)$ representa o custo de realizar a tarefa t_i.

Depende diretamente do conceito de somatório, pois frequentemente queremos minimizar o custo total de um conjunto de escolhas. Se $S \subseteq A$ é um subconjunto de elementos escolhidos, o custo total associado a S é dado por:

$$C(S) = \sum_{a \in S} c(a).$$

Função c-disjunta Uma **função c-disjunta** é uma função $f : A \to B$ que, para quaisquer $a_1, a_2 \in A$ com $a_1 \neq a_2$, as imagens $f(a_1)$ e $f(a_2)$ são disjuntas, ou seja, $f(a_1) \cap f(a_2) = \varnothing$. Em outras palavras, elementos distintos do domínio são mapeados para conjuntos disjuntos no contradomínio.

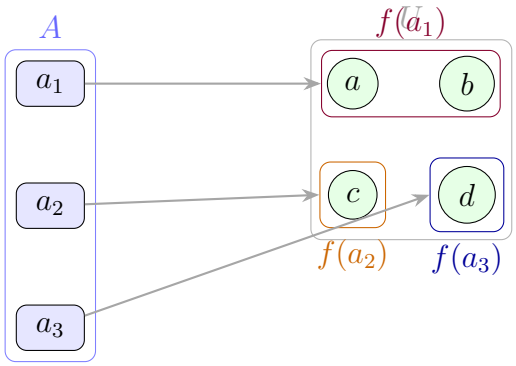


Figura 18: Função c-disjunta: cada $a \in A$ mapeia para um *subconjunto* de U, e as imagens são dois a dois disjuntas (sem sobreposição). No exemplo, $f(a_1) = \{a, b\}$, $f(a_2) = \{c\}$ e $f(a_3) = \{d\}$.

Exemplo: Considere $A = \{1, 2, 3\}$ e $B = \{\{a\}, \{b\}, \{c\}, \{d\}\}$. Definimos a função $f : A \to B$ como:

$$f(1) = \{a, b\}, \quad f(2) = \{c\}, \quad f(3) = \{d\}.$$

Aqui, f é c-disjunta, pois $f(1) \cap f(2) = \varnothing$, $f(1) \cap f(3) = \varnothing$ e $f(2) \cap f(3) = \varnothing$.

Função c-viável

Uma **função c-viável** é uma função $g : A \to \mathbb{R}^+$ que satisfaz certas condições de viabilidade relacionadas a um conjunto de restrições ou critérios. Essas funções são frequentemente usadas em otimização e teoria dos grafos para garantir que as soluções propostas atendam a requisitos específicos.

Exemplo: Considere um conjunto de projetos $P = \{p_1, p_2, p_3\}$ e uma função $g : P \to \mathbb{R}^+$ que atribui um valor de viabilidade a cada projeto. Suponha que temos a restrição de que a soma dos valores de viabilidade deve ser menor ou igual a um certo limite L. Se definirmos:

$$g(p_1) = 4, \quad g(p_2) = 6, \quad g(p_3) = 3,$$

então a função g é c-viável se $g(p_1) + g(p_2) + g(p_3) \leq L$.

Essas funções são essenciais para garantir que as soluções propostas em problemas de otimização sejam práticas e atendam aos critérios estabelecidos.

Funções de otimização

Do ponto de vista semiótico, "melhor" exprime uma preferência entre interpretações: ao comparar alternativas, escolhemos aquela cuja significação é mais adequada a um critério. Para tornar isso operacional, a matemática troca "fazer mais sentido" por "ter maior (ou menor) valor" em uma escala formal: fixamos (i) um conjunto de soluções viáveis $\mathcal{F}$ e (ii) uma função numérica sobre $\mathcal{F}$ que induz uma ordem de comparação.

Formalmente, usamos uma **função objetivo** (ou **função de otimização**)

$$h : \mathcal{F} \to \mathbb{R},$$

que atribui um número real a cada solução. Buscamos uma solução $S^* \in \mathcal{F}$ que *minimize* ou *maximize* h (isto é, um argmin ou argmax). Quando esse número resulta da soma de contribuições elementares, obtemos o caso aditivo, em ligação direta com os somatórios apresentados antes.

Caso *aditivo*. Quando cada elemento $a \in A$ tem um custo $c(a) \geq 0$ e as soluções são subconjuntos $S \subseteq A$, a função objetivo mais comum é o custo total

$$C(S) = \sum_{a \in S} c(a),$$

que desejamos *minimizar*. De modo análogo, se cada item tem um benefício $p(a) \geq 0$, podemos *maximizar* o benefício total $P(S) = \sum_{a \in S} p(a)$, possivelmente sujeito a restrições (por exemplo, de orçamento ou capacidade).

Exemplo acessível. Considere produtos $X = \{x_1, x_2, x_3\}$ com lucro $p(x_1) = 10$, $p(x_2) = 15$, $p(x_3) = 7$. Se houver um limite de custo que impede escolher todos, o objetivo típico é escolher um subconjunto $S \subseteq X$ que maximize $\sum_{x \in S} p(x)$ respeitando as restrições. Essa forma reflete exatamente os somatórios introduzidos antes.

Em grafos, a função objetivo costuma somar pesos sobre uma estrutura escolhida:

- *Caminho mais curto* de s a t: minimizar $\sum_{e \in P} c(e)$ entre os caminhos P de s até t.
- *Árvore geradora mínima*: minimizar $\sum_{e \in T} c(e)$ entre as árvores geradoras T do grafo.
- *Fluxo máximo* (em dígrafos com capacidades e vértices fonte s e sumidouro t): enviar o máximo possível de s para t sem ultrapassar as capacidades; o *valor do fluxo* é a soma que sai de s (igual ao que chega em t). Veremos adiante que esse ótimo coincide com o valor de um corte s–t mínimo.

Em todos os casos, a função objetivo explicita o critério de "melhor", e as restrições determinam quais soluções são aceitáveis — dois ingredientes que retomaremos mais adiante.

2.2.4 Otimização

3 Considerações Finais