AED2 - Algoritmos e Estr. de Dados II

Aula 25: Aplicações de Fluxo Máximo

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

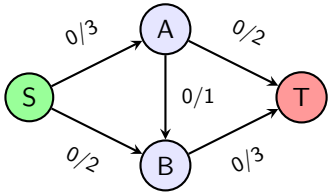
CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Junho de 2026

Sumário

1. O Coração de Tudo: Redes de Fluxo

Lembram do **Fluxo Máximo**? Queremos transportar o máximo possível de "material" de uma Fonte (S) para um Sorvedouro (T) respeitando as **capacidades** das arestas (canos).

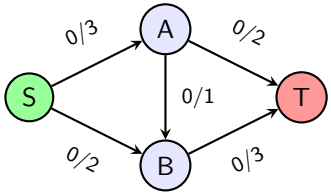


1. O Coração de Tudo: Redes de Fluxo

Lembram do **Fluxo Máximo**? Queremos transportar o máximo possível de "material" de uma Fonte (S) para um Sorvedouro (T) respeitando as **capacidades** das arestas (canos).

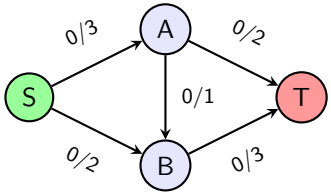
Conservação de Fluxo

Para todo vértice (exceto S e T):
$\sum$ Fluxo Entrando $= \sum$ Fluxo Saindo



1. O Coração de Tudo: Redes de Fluxo

Lembram do **Fluxo Máximo**? Queremos transportar o máximo possível de "material" de uma Fonte (S) para um Sorvedouro (T) respeitando as **capacidades** das arestas (canos).



Conservação de Fluxo

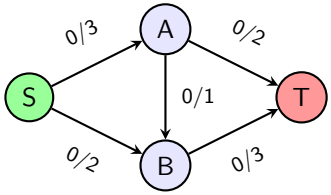
Para todo vértice (exceto S e T):
$\sum$ Fluxo Entrando $= \sum$ Fluxo Saindo

Algoritmos para Fluxo Máximo

- **Ford-Fulkerson (DFS):** $O(E \cdot f^*)$
- **Edmonds-Karp (BFS):** $O(V \cdot E^2)$
- **Dinic (Grafo de Níveis):** $O(V^2 \cdot E)$

1. O Coração de Tudo: Redes de Fluxo

Lembram do **Fluxo Máximo**? Queremos transportar o máximo possível de "material" de uma Fonte (S) para um Sorvedouro (T) respeitando as **capacidades** das arestas (canos).



Conservação de Fluxo

Para todo vértice (exceto S e T):
$\sum$ Fluxo Entrando $= \sum$ Fluxo Saindo

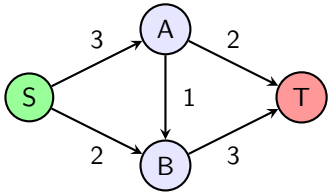
Algoritmos para Fluxo Máximo

- **Ford-Fulkerson (DFS):** $O(E \cdot f^*)$
- **Edmonds-Karp (BFS):** $O(V \cdot E^2)$
- **Dinic (Grafo de Níveis):** $O(V^2 \cdot E)$

Atenção: Calcular fluxo é apenas 10% do trabalho em ICPC/OBI. Os 90% restantes são **reduzir** o problema a um fluxo!

Corte (Cut)

Um corte (S_{cut}, T_{cut}) particiona os vértices do grafo em dois conjuntos disjuntos, onde $S \in S_{cut}$ e $T \in T_{cut}$. A **Capacidade do Corte** é a soma das capacidades das arestas que vão de S_{cut} para T_{cut}.



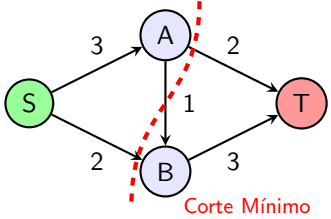
2. Teorema Min-Cut Max-Flow (Ford & Fulkerson, 1956)

Corte (Cut)

Um corte (S_{cut}, T_{cut}) particiona os vértices do grafo em dois conjuntos disjuntos, onde $S \in S_{cut}$ e $T \in T_{cut}$. A **Capacidade do Corte** é a soma das capacidades das arestas que vão de S_{cut} para T_{cut}.

O Teorema

O valor do **Fluxo Máximo** de S a T é exatamente igual à capacidade do **Corte Mínimo** que separa S de T.



Arestas cortadas (frente): $A \to T$ (2) e $S \to B$ (2).
Capacidade do Corte $= 2 + 2 = 4$.
Portanto, Fluxo Máx $= 4$!

2. Teorema Min-Cut Max-Flow (Ford & Fulkerson, 1956)

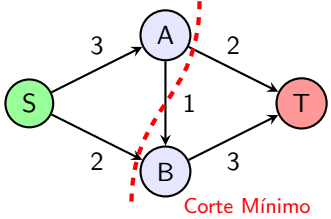
Corte (Cut)

Um corte (S_{cut}, T_{cut}) particiona os vértices do grafo em dois conjuntos disjuntos, onde $S \in S_{cut}$ e $T \in T_{cut}$. A **Capacidade do Corte** é a soma das capacidades das arestas que vão de S_{cut} para T_{cut}.

O Teorema

O valor do **Fluxo Máximo** de S a T é exatamente igual à capacidade do **Corte Mínimo** que separa S de T.

Significado Prático: Se o fluxo máximo é 10, o "gargalo" da rede (as arestas que, se removidas, desconectam S de T) tem soma de capacidades

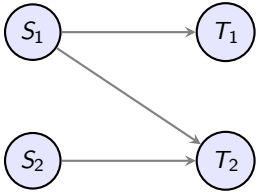


Arestas cortadas (frente): $A \to T$ (2) e $S \to B$ (2).
Capacidade do Corte $= 2 + 2 = 4$.
Portanto, Fluxo Máx $= 4$!

3. Truque 1: Múltiplas Fontes e Múltiplos Sorvedouros

E se o problema tiver 3 fábricas (S_1, S_2, S_3) e 4 lojas (T_1, T_2, T_3, T_4)?

A Solução do Super-Nó

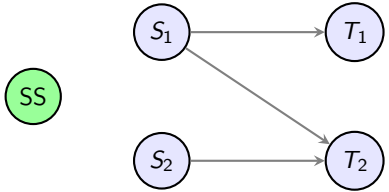


3. Truque 1: Múltiplas Fontes e Múltiplos Sorvedouros

E se o problema tiver 3 fábricas (S_1, S_2, S_3) e 4 lojas (T_1, T_2, T_3, T_4)?

A Solução do Super-Nó

1. Crie um **Super-Source** (Super-Fonte) virtual SS.

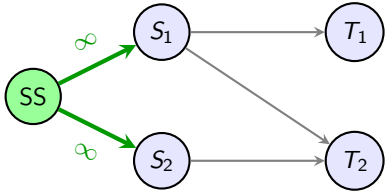


3. Truque 1: Múltiplas Fontes e Múltiplos Sorvedouros

E se o problema tiver 3 fábricas (S_1, S_2, S_3) e 4 lojas (T_1, T_2, T_3, T_4)?

A Solução do Super-Nó

1. Crie um **Super-Source** (Super-Fonte) virtual SS.
2. Conecte SS a cada fonte real com uma aresta de capacidade ∞ (ou com a produção máxima daquela fábrica).

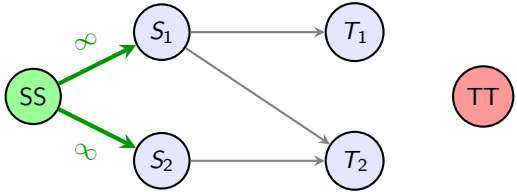


3. Truque 1: Múltiplas Fontes e Múltiplos Sorvedouros

E se o problema tiver 3 fábricas (S_1, S_2, S_3) e 4 lojas (T_1, T_2, T_3, T_4)?

A Solução do Super-Nó

1. Crie um **Super-Source** (Super-Fonte) virtual SS.
2. Conecte SS a cada fonte real com uma aresta de capacidade ∞ (ou com a produção máxima daquela fábrica).
3. Crie um **Super-Sink** (Super-Sorv.) virtual TT.

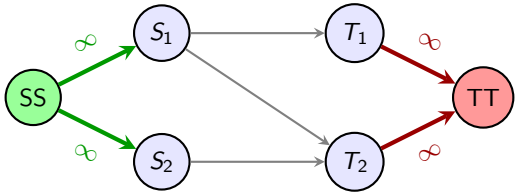


3. Truque 1: Múltiplas Fontes e Múltiplos Sorvedouros

E se o problema tiver 3 fábricas (S_1, S_2, S_3) e 4 lojas (T_1, T_2, T_3, T_4)?

A Solução do Super-Nó

1. Crie um **Super-Source** (Super-Fonte) virtual SS.
2. Conecte SS a cada fonte real com uma aresta de capacidade ∞ (ou com a produção máxima daquela fábrica).
3. Crie um **Super-Sink** (Super-Sorv.) virtual TT.
4. Conecte cada sorvedouro real a TT com capacidade ∞ (ou demanda).

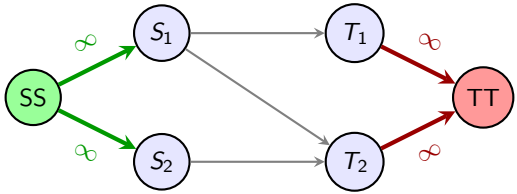


3. Truque 1: Múltiplas Fontes e Múltiplos Sorvedouros

E se o problema tiver 3 fábricas (S_1, S_2, S_3) e 4 lojas (T_1, T_2, T_3, T_4)?

A Solução do Super-Nó

1. Crie um **Super-Source** (Super-Fonte) virtual SS.
2. Conecte SS a cada fonte real com uma aresta de capacidade ∞ (ou com a produção máxima daquela fábrica).
3. Crie um **Super-Sink** (Super-Sorv.) virtual TT.
4. Conecte cada sorvedouro real a TT com capacidade ∞ (ou demanda).

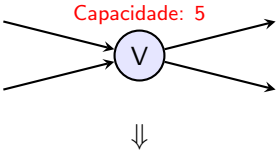


Resultado: O problema se reduz ao Fluxo Máximo clássico de SS para TT!

4. Truque 2: Capacidade em Vértices

Algoritmos de Fluxo limitam o quanto passa nas **arestas**. E se uma **cidade (vértice)** só puder processar C unidades de material por dia?

A Solução de Split (Vértice Divisão)

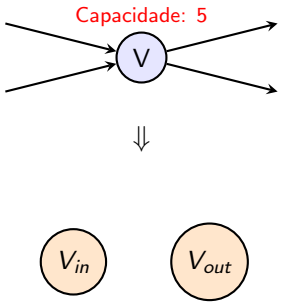


4. Truque 2: Capacidade em Vértices

Algoritmos de Fluxo limitam o quanto passa nas **arestas**. E se uma **cidade (vértice)** só puder processar C unidades de material por dia?

A Solução de Split (Vértice Divisão)

Dividimos o vértice V em dois vértices virtuais: V_{in} e V_{out}.



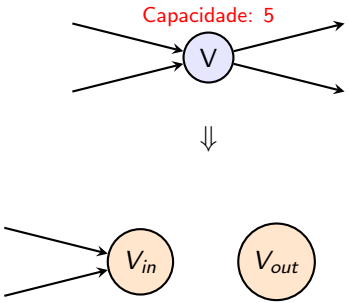
4. Truque 2: Capacidade em Vértices

Algoritmos de Fluxo limitam o quanto passa nas **arestas**. E se uma **cidade (vértice)** só puder processar C unidades de material por dia?

A Solução de Split (Vértice Divisão)

Dividimos o vértice V em dois vértices virtuais: V_{in} e V_{out}.

- Todas as arestas que **entravam** em V agora entram em V_{in}.



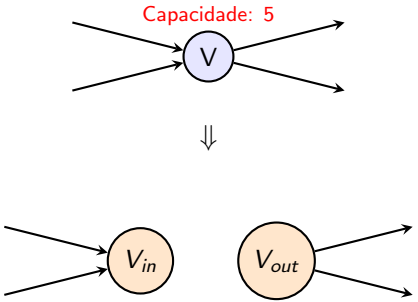
4. Truque 2: Capacidade em Vértices

Algoritmos de Fluxo limitam o quanto passa nas **arestas**. E se uma **cidade (vértice)** só puder processar C unidades de material por dia?

A Solução de Split (Vértice Divisão)

Dividimos o vértice V em dois vértices virtuais: V_{in} e V_{out}.

- Todas as arestas que **entravam** em V agora entram em V_{in}.
- Todas as arestas que **saíam** de V agora saem de V_{out}.



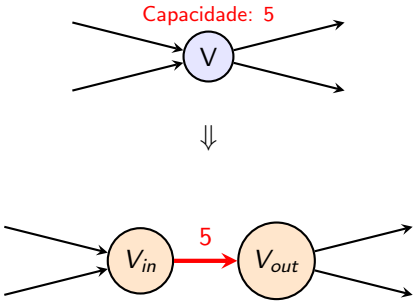
4. Truque 2: Capacidade em Vértices

Algoritmos de Fluxo limitam o quanto passa nas **arestas**. E se uma **cidade (vértice)** só puder processar C unidades de material por dia?

A Solução de Split (Vértice Divisão)

Dividimos o vértice V em dois vértices virtuais: V_{in} e V_{out}.

- Todas as arestas que **entravam** em V agora entram em V_{in}.
- Todas as arestas que **saíam** de V agora saem de V_{out}.
- Adicionamos uma **única aresta** de V_{in} para V_{out} com a capacidade C do vértice.



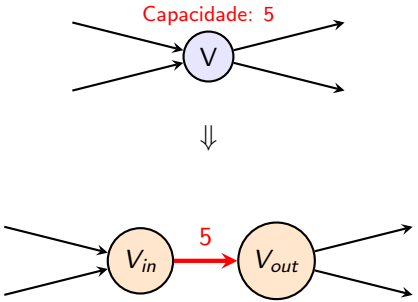
4. Truque 2: Capacidade em Vértices

Algoritmos de Fluxo limitam o quanto passa nas **arestas**. E se uma **cidade (vértice)** só puder processar C unidades de material por dia?

A Solução de Split (Vértice Divisão)

Dividimos o vértice V em dois vértices virtuais: V_{in} e V_{out}.

- Todas as arestas que **entravam** em V agora entram em V_{in}.
- Todas as arestas que **saíam** de V agora saem de V_{out}.
- Adicionamos uma **única aresta** de V_{in} para V_{out} com a capacidade C do vértice.



Por que funciona? Qualquer fluxo passando por V será forçado a passar pela aresta interna

5. Truque 3: Arestas Bidirecionais

Se a estrada entre A e B é de mão dupla e tem capacidade 10 (total somando as duas direções ou 10 para cada lado)?

Caso A: 10 para cada lado (Independentes)

Basta adicionar duas arestas direcionadas separadas na rede residual.

- $A \to B$ com cap=10
- $B \to A$ com cap=10

No Algoritmo de Dinic ou Edmonds-Karp, adicione as arestas reversas padrão (back-edges) para cada uma delas (com capacidade inicial 0).

5. Truque 3: Arestas Bidirecionais

Se a estrada entre A e B é de mão dupla e tem capacidade 10 (total somando as duas direções ou 10 para cada lado)?

Caso A: 10 para cada lado (Independentes)

Basta adicionar duas arestas direcionadas separadas na rede residual.

- $A \to B$ com cap=10
- $B \to A$ com cap=10

No Algoritmo de Dinic ou Edmonds-Karp, adicione as arestas reversas padrão (back-edges) para cada uma delas (com capacidade inicial 0).

Caso B: 10 de capacidade total

Se o cano passa até 10 litros independentemente da direção, o limite **global** do cano é 10.

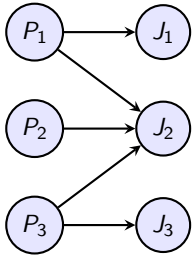
- O algoritmo já lida com o cancelamento de fluxo!
- Para modelar aresta não-direcional $A - B$ com capacidade C, basta criá-la assim:
- `Edge(A, B, cap=C, rev_cap=C)`

Quando você empurrar fluxo de $A \to B$, a capacidade residual $B \to A$ aumentará

6. Aplicação 1: Emparelhamento Bipartido Máximo

O Problema

Temos um conjunto de Trabalhadores e de Tarefas. Cada trabalhador sabe fazer algumas tarefas (grafo bipartido). Cada trabalhador só pode fazer **uma** tarefa, e cada tarefa só precisa de **um** trabalhador. Como emparelhar o **máximo** de pessoas às tarefas?



6. Aplicação 1: Emparelhamento Bipartido Máximo

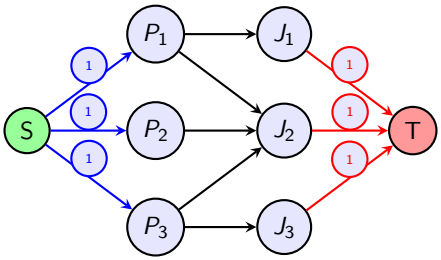
O Problema

Temos um conjunto de Trabalhadores e de Tarefas. Cada trabalhador sabe fazer algumas tarefas (grafo bipartido). Cada trabalhador só pode fazer **uma** tarefa, e cada tarefa só precisa de **um** trabalhador. Como emparelhar o **máximo** de pessoas às tarefas?

Modelagem de Fluxo:

1. Crie S e conecte a todos os Trabalhadores (cap=1).
2. Crie T e conecte todas as Tarefas a T (cap=1).
3. Conecte Trabalhador $\rightarrow$ Tarefa que ele sabe (cap=1).
4. Rode Fluxo Máximo!

Intuição: Como as capacidades são 1, cada unidade de fluxo de S a T forma um



7. Extensão do Emparelhamento (Matching com Limites)

E se a empresa tiver computadores (Tarefas) e engenheiros (Pessoas).

- Cada pessoa precisa de **exatamente 1** PC.
- Mas cada PC pode ser usado por **até 3** engenheiros (turnos diferentes).
- Certas pessoas não podem usar certos PCs (distância, restrição de software).

7. Extensão do Emparelhamento (Matching com Limites)

E se a empresa tiver computadores (Tarefas) e engenheiros (Pessoas).

- Cada pessoa precisa de **exatamente 1** PC.
- Mas cada PC pode ser usado por **até 3** engenheiros (turnos diferentes).
- Certas pessoas não podem usar certos PCs (distância, restrição de software).

Solução Elegante

Você não precisa recriar a roda ou duplicar vértices de computadores! Basta mudar **UMA capacidade** na modelagem:

- $S \rightarrow$ Engenheiro: capacidade **1** (o engenheiro só pega 1 PC).
- Engenheiro $\rightarrow$ PC possível: capacidade **1**.
- PC $\rightarrow T$: capacidade **3** (este nó/PC pode escoar até 3 unidades de fluxo).

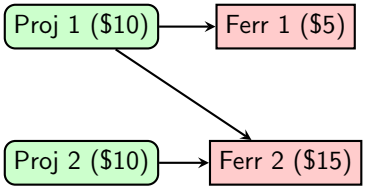
Rodamos Max-Flow e pronto. Se o Fluxo Máximo for igual ao número total de Engenheiros, todos conseguiram um PC respeitando as regras.

8. Aplicação 2: O Problema da Seleção de Projetos

Uma empresa tem projetos independentes.

- Cada **Projeto** rende um **Lucro** P_i.
- Cada projeto depende de uma ou mais **Ferramentas** para ser feito.
- Cada **Ferramenta** tem um **Custo** C_j para ser comprada (compra-se 1 vez).

Objetivo: Escolher quais projetos fazer para **maximizar** Lucro - Custo.



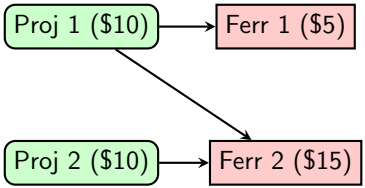
8. Aplicação 2: O Problema da Seleção de Projetos

Uma empresa tem projetos independentes.

- Cada **Projeto** rende um **Lucro** P_i.
- Cada projeto depende de uma ou mais **Ferramentas** para ser feito.
- Cada **Ferramenta** tem um **Custo** C_j para ser comprada (compra-se 1 vez).

Objetivo: Escolher quais projetos fazer para **maximizar** Lucro - Custo.

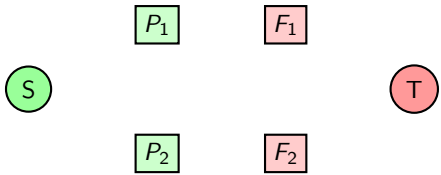
O Desafio: Se um projeto rende 10, mas a ferramenta custa 15, eu não faço o projeto. **Porém**, se **dois** projetos renderem 10 cada e usarem a mesma ferramenta de 15, vale a pena (20 - 15 = 5)! Isso quebra heurísticas gulosas normais.



9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

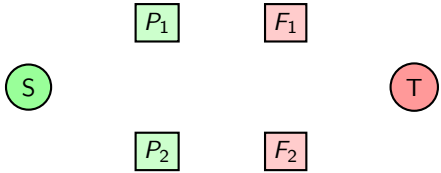


9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

1. Crie S e T.

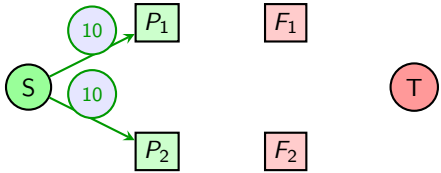


9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

1. Crie S e T.
2. Aresta $S \to P_i$ com cap = Lucro do Projeto.

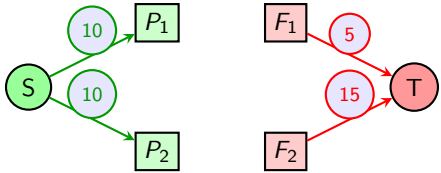


9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

1. Crie S e T.
2. Aresta $S \to P_i$ com cap = Lucro do Projeto.
3. Aresta $F_j \to T$ com cap = Custo da Ferramenta.

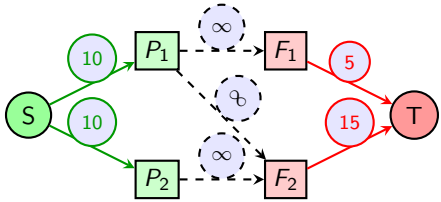


9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

1. Crie S e T.
2. Aresta $S \to P_i$ com cap = Lucro do Projeto.
3. Aresta $F_j \to T$ com cap = Custo da Ferramenta.
4. Aresta $P_i \to F_j$ (dependência) com cap $= \infty$.

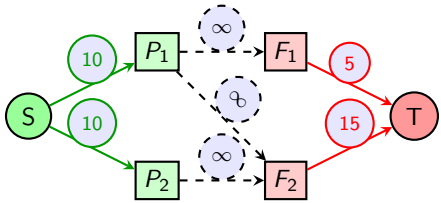


9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

1. Crie S e T.
2. Aresta $S \to P_i$ com cap = Lucro do Projeto.
3. Aresta $F_j \to T$ com cap = Custo da Ferramenta.
4. Aresta $P_i \to F_j$ (dependência) com cap $= \infty$.



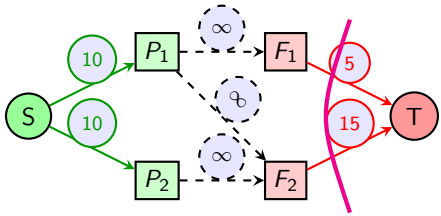
Significado do Corte: Se o algoritmo cortar a aresta $S \to P_i$, significa "Não fazer P_i" (perdeu o lucro). Se cortar $F_j \to T$, significa "Comprar F_j" (pagou o custo). Arestas ∞

9. Modelando a Seleção de Projetos via Min-Cut

Construímos a rede de fluxo para achar o **menor prejuízo** (Min-Cut). O Lucro Líquido Máximo = (Soma de todos os lucros de projetos) - (Min-Cut).

Modelagem

1. Crie S e T.
2. Aresta $S \to P_i$ com cap = Lucro do Projeto.
3. Aresta $F_j \to T$ com cap = Custo da Ferramenta.
4. Aresta $P_i \to F_j$ (dependência) com cap $= \infty$.



Min-Cut = 20 (compra F1, F2)

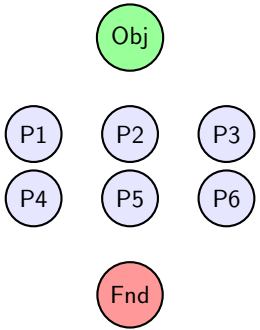
Significado do Corte: Se o algoritmo cortar a aresta $S \to P_i$, significa "Não fazer P_i" (perdeu o lucro). Se cortar $F_j \to T$, significa "Comprar F_j" (pagou o custo). Arestas ∞

Cálculo: Soma Lucros (20) - Min-Cut (20) = **0**. Se P_1

10. Aplicação 3: Segmentação de Imagens (Visão Computacional)

Como o Photoshop remove o fundo de uma imagem automaticamente? Ou imagens médicas segmentam um tumor?

A técnica interativa "Graph Cut" trata cada **pixel** como um vértice.

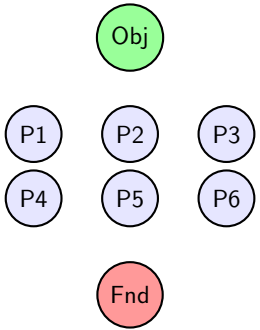


10. Aplicação 3: Segmentação de Imagens (Visão Computacional)

Como o Photoshop remove o fundo de uma imagem automaticamente? Ou imagens médicas segmentam um tumor?

A técnica interativa "Graph Cut" trata cada **pixel** como um vértice.

1. Crie S (representando o Objeto/Foreground).

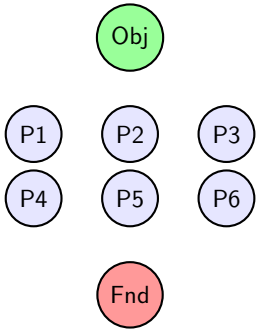


10. Aplicação 3: Segmentação de Imagens (Visão Computacional)

Como o Photoshop remove o fundo de uma imagem automaticamente? Ou imagens médicas segmentam um tumor?

A técnica interativa "Graph Cut" trata cada **pixel** como um vértice.

1. Crie S (representando o Objeto/Foreground).
2. Crie T (representando o Fundo/Background).

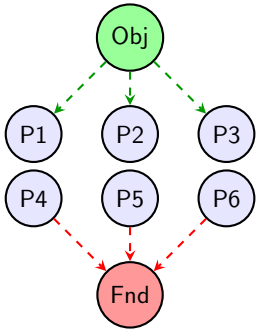


10. Aplicação 3: Segmentação de Imagens (Visão Computacional)

Como o Photoshop remove o fundo de uma imagem automaticamente? Ou imagens médicas segmentam um tumor?

A técnica interativa "Graph Cut" trata cada **pixel** como um vértice.

1. Crie S (representando o Objeto/Foreground).
2. Crie T (representando o Fundo/Background).
3. **Arestas de Probabilidade:** $S \rightarrow Pixel$ (prob. de ser objeto, cor vermelha, ex). $Pixel \rightarrow T$ (prob. de ser fundo).

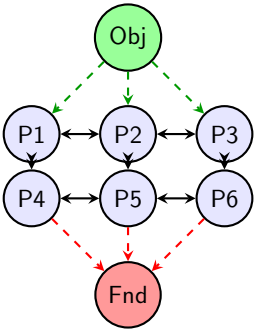


10. Aplicação 3: Segmentação de Imagens (Visão Computacional)

Como o Photoshop remove o fundo de uma imagem automaticamente? Ou imagens médicas segmentam um tumor?

A técnica interativa "Graph Cut" trata cada **pixel** como um vértice.

1. Crie S (representando o Objeto/Foreground).
2. Crie T (representando o Fundo/Background).
3. **Arestas de Probabilidade:** $S \rightarrow Pixel$ (prob. de ser objeto, cor vermelha, ex). $Pixel \rightarrow T$ (prob. de ser fundo).
4. **Arestas Espaciais:** Pixel vizinho $\leftrightarrow$ Pixel vizinho com alta capacidade se tiverem **cores parecidas**.

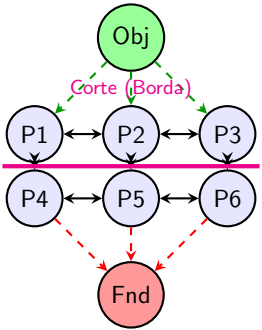


10. Aplicação 3: Segmentação de Imagens (Visão Computacional)

Como o Photoshop remove o fundo de uma imagem automaticamente? Ou imagens médicas segmentam um tumor?

A técnica interativa "Graph Cut" trata cada **pixel** como um vértice.

1. Crie S (representando o Objeto/Foreground).
2. Crie T (representando o Fundo/Background).
3. **Arestas de Probabilidade:** $S \rightarrow Pixel$ (prob. de ser objeto, cor vermelha, ex). $Pixel \rightarrow T$ (prob. de ser fundo).
4. **Arestas Espaciais:** Pixel vizinho $\leftrightarrow$ Pixel vizinho com alta capacidade se tiverem **cores parecidas**.



O Corte Mínimo

Rodando Max-Flow, o algoritmo vai cortar as

11. Aplicação 4: Roteamento de Aviões / Escalonamento

O Problema (Estilo Maratona)

Uma empresa aérea tem uma lista de voos diários. Cada voo tem uma **Origem**, **Destino**, **Hora de Partida** e **Hora de Chegada**. Para um avião fazer o Voo A e depois o Voo B, ele precisa chegar de A e ter tempo ($+t_{prep}$) de voar vazio para a origem de B antes de B decolar. Qual o **número MÍNIMO de aviões** necessários para cobrir todos os voos?

11. Aplicação 4: Roteamento de Aviões / Escalonamento

O Problema (Estilo Maratona)

Uma empresa aérea tem uma lista de voos diários. Cada voo tem uma **Origem**, **Destino**, **Hora de Partida** e **Hora de Chegada**. Para um avião fazer o Voo A e depois o Voo B, ele precisa chegar de A e ter tempo ($+t_{prep}$) de voar vazio para a origem de B antes de B decolar. Qual o **número MÍNIMO de aviões** necessários para cobrir todos os voos?

Modelagem com Cobertura Mínima de Caminhos (DAG):

1. Cada Voo é um **Vértice**.
2. Se é possível o mesmo avião fazer o voo A e depois o B, criamos uma aresta $A \rightarrow B$.
3. Este grafo é Acíclico (DAG) por causa do tempo.

11. Aplicação 4: Roteamento de Aviões / Escalonamento

O Problema (Estilo Maratona)

Uma empresa aérea tem uma lista de voos diários. Cada voo tem uma **Origem**, **Destino**, **Hora de Partida** e **Hora de Chegada**. Para um avião fazer o Voo A e depois o Voo B, ele precisa chegar de A e ter tempo ($+t_{prep}$) de voar vazio para a origem de B antes de B decolar. Qual o **número MÍNIMO de aviões** necessários para cobrir todos os voos?

Modelagem com Cobertura Mínima de Caminhos (DAG):

1. Cada Voo é um **Vértice**.
2. Se é possível o mesmo avião fazer o voo A e depois o B, criamos uma aresta $A \rightarrow B$.
3. Este grafo é Acíclico (DAG) por causa do tempo.

O Truque do Fluxo

Usamos **Bipartite Matching** para achar o número máximo de "emendas" (um avião pegando 2 voos seguidos). Cada emenda salva 1 avião novo que teríamos que comprar. **Resposta:** V (Total de Voos) - Max Matching.

12. Aplicação 5: Redução com Conservação de Nódos

Problema da Tabela Dinâmica

É dada a **soma das linhas** e a **soma das colunas** de uma matriz $N \times M$. Além disso, cada célula (i, j) pode ter um valor máximo $C_{i,j}$ (alguns podem ser zero). É possível construir essa matriz de forma válida?

12. Aplicação 5: Redução com Conservação de Nódos

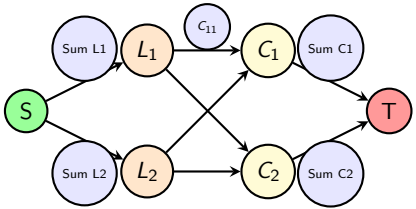
Problema da Tabela Dinâmica

É dada a **soma das linhas** e a **soma das colunas** de uma matriz $N \times M$. Além disso, cada célula (i, j) pode ter um valor máximo $C_{i,j}$ (alguns podem ser zero). É possível construir essa matriz de forma válida?

Modelagem de Fluxo:

- $S \rightarrow \text{Linha}_i$ (cap = soma exigida da Linha i).
- $\text{Linha}_i \rightarrow \text{Coluna}_j$ (cap = $C_{i,j}$, máximo permitido na célula).
- $\text{Coluna}_j \rightarrow T$ (cap = soma exigida da Coluna j).

Verificação: Se o **Fluxo Máximo** for igual à soma total esperada de todas as linhas (que deve bater com a das colunas), a matriz **existe**. O fluxo na aresta Linha $\rightarrow$ Coluna é o valor final da célula!



13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

1. Precisa **emparelhar** ou designar elementos de 2 ou 3 conjuntos respeitando limites de quantos cada um aguenta (Pessoas $\rightarrow$ Máquinas, Trens $\rightarrow$ Horários).

13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

1. Precisa **emparelhar** ou designar elementos de 2 ou 3 conjuntos respeitando limites de quantos cada um aguenta (Pessoas $\rightarrow$ Máquinas, Trens $\rightarrow$ Horários).
2. Precisa **separar** um grafo em duas partes com o menor prejuízo/custo (Problemas com matrizes, zonas de controle, segmentação de áreas) $\rightarrow$ **Min-Cut**.

13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

1. Precisa **emparelhar** ou designar elementos de 2 ou 3 conjuntos respeitando limites de quantos cada um aguenta (Pessoas $\rightarrow$ Máquinas, Trens $\rightarrow$ Horários).
2. Precisa **separar** um grafo em duas partes com o menor prejuízo/custo (Problemas com matrizes, zonas de controle, segmentação de áreas) $\rightarrow$ **Min-Cut**.
3. Há uma rede física de tráfego, tubulação, telecomunicação, onde arestas são gargalos e precisa maximizar o envio (*O clássico!*).

13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

1. Precisa **emparelhar** ou designar elementos de 2 ou 3 conjuntos respeitando limites de quantos cada um aguenta (Pessoas $\rightarrow$ Máquinas, Trens $\rightarrow$ Horários).
2. Precisa **separar** um grafo em duas partes com o menor prejuízo/custo (Problemas com matrizes, zonas de controle, segmentação de áreas) $\rightarrow$ **Min-Cut**.
3. Há uma rede física de tráfego, tubulação, telecomunicação, onde arestas são gargalos e precisa maximizar o envio (*O clássico!*).
4. Problemas de dependência onde fazer algo gera lucro, mas requer pagar por ferramentas/pré-requisitos.

13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

1. Precisa **emparelhar** ou designar elementos de 2 ou 3 conjuntos respeitando limites de quantos cada um aguenta (Pessoas $\rightarrow$ Máquinas, Trens $\rightarrow$ Horários).
2. Precisa **separar** um grafo em duas partes com o menor prejuízo/custo (Problemas com matrizes, zonas de controle, segmentação de áreas) $\rightarrow$ **Min-Cut**.
3. Há uma rede física de tráfego, tubulação, telecomunicação, onde arestas são gargalos e precisa maximizar o envio (*O clássico!*).
4. Problemas de dependência onde fazer algo gera lucro, mas requer pagar por ferramentas/pré-requisitos.
5. Recuperar uma matriz ou grid satisfazendo as somas marginais.

13. O Sumário do Engenheiro Competitivo

Se você vir um problema com as seguintes características, pense em **Fluxo Máximo**:

1. Precisa **emparelhar** ou designar elementos de 2 ou 3 conjuntos respeitando limites de quantos cada um aguenta (Pessoas $\rightarrow$ Máquinas, Trens $\rightarrow$ Horários).
2. Precisa **separar** um grafo em duas partes com o menor prejuízo/custo (Problemas com matrizes, zonas de controle, segmentação de áreas) $\rightarrow$ **Min-Cut**.
3. Há uma rede física de tráfego, tubulação, telecomunicação, onde arestas são gargalos e precisa maximizar o envio (*O clássico!*).
4. Problemas de dependência onde fazer algo gera lucro, mas requer pagar por ferramentas/pré-requisitos.
5. Recuperar uma matriz ou grid satisfazendo as somas marginais.

O verdadeiro teste

Construir o grafo com Fonte, Sorvedouro, Vértices Virtuais e Capacidades Corretas é **a parte difícil**. O código do algoritmo (Dinic/Edmonds-Karp) é só uma caixa preta!