Projeto e Análise de Algoritmos II (MC558)
Caminhos mínimos

Prof. Dr. Ruben Interian

Resumo

Resumo

Revisão do conteúdo

- O problema dos **caminhos mínimos de fonte única** pode ser resolvido eficientemente quando não há ciclos de peso negativo.
- Os algoritmos baseados em relaxação usam os métodos **InitializeSingleSource** e **Relax** para construir as distâncias d e a árvore de caminhos mínimos π.
- O **algoritmo de Dijkstra** resolve o Problema dos Caminhos Mínimos de fonte única quando não temos arcos de peso negativo. Quando temos arcos negativos, o algoritmo de Dijkstra não funciona.

Objetivo

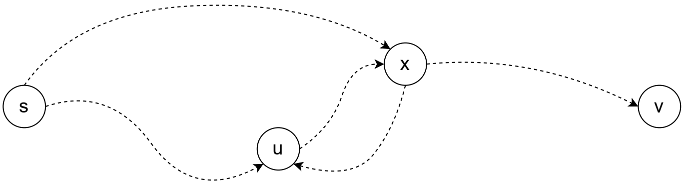
Resolver o problema de encontrar os **caminhos mínimos de fonte única** mesmo quando há arcos de peso negativo (mas não há ciclos negativos): algoritmo de Bellman-Ford.

Resumo

Arcos negativos e ciclos negativos

- Todos os algoritmos que vimos (baseados em relaxação) usam a propriedade da **subestrutura ótima de caminhos mínimos**.
- Se temos arcos negativos, é possível que existam **ciclos negativos** no grafo.
- A propriedade da subestrutura ótima **vale para grafos direcionados que não contêm ciclos de custo negativo**.

Subestrutura ótima de caminhos mínimos: intuição (retrospectiva)



Vamos supor que o caminho P de s a v é mínimo, mas o subcaminho de s a u não é. Então existe um caminho de s a u de peso menor. Veja que ele precisa passar por P (caso contrário, haveria também um caminho menor de s a v).
Seja um vértice x que está nesse caminho e em P.
Existe um ciclo que contém os vértices u e x. O seu custo é **negativo** (por quê?).

A propriedade da subestrutura ótima vale para grafos direcionados que não contêm ciclos de custo negativo.

Arcos negativos e ciclos negativos

- O problema dos caminhos mínimos para instâncias com ciclos negativos é **NP-difícil**.
- Até agora, **ninguém conseguiu** encontrar um algoritmo eficiente para resolver nenhum problema NP-difícil. Por este motivo, vamos nos restringir ao Problema de Caminhos Mínimos sem ciclos negativos.
- Futuramente durante o curso veremos o significado dos conceitos: **problema NP**, **NP-completo** e **NP-difícil**.

Algoritmo de Bellman-Ford

- Um algoritmo que resolve o problema dos caminhos mínimos quando há arcos de peso negativo, mas não ciclos negativos, é o algoritmo de **Bellman-Ford**.
- **Bellman-Ford** resolve o problema **mesmo se houver ciclos negativos (!)** (desde que não sejam alcançáveis por s).
- É mais um algoritmo baseado em **relaxação**.

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k), **nessa ordem**.

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k), **nessa ordem**.

1. Se executamos **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1) estaria relaxado.

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k), **nessa ordem**.

1. Se executamos **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1) estaria relaxado.
2. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2) estariam relaxados, **nessa ordem**.

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k), **nessa ordem**.

1. Se executamos **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1) estaria relaxado.
2. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2) estariam relaxados, **nessa ordem**.
3. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2), (v_2, v_3) estariam relaxados, **nessa ordem**.

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), . . . , (v_{k-1}, v_k), **nessa ordem**.

1. Se executamos **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1) estaria relaxado.
2. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2) estariam relaxados, **nessa ordem**.
3. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2), (v_2, v_3) estariam relaxados, **nessa ordem**.

   Repetimos $|V| - 1$ vezes . . . (por quê $|V| - 1$?)

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k), **nessa ordem**.

1. Se executamos **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1) estaria relaxado.
2. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2) estariam relaxados, **nessa ordem**.
3. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2), (v_2, v_3) estariam relaxados, **nessa ordem**.

   Repetimos $|V| - 1$ vezes $\ldots$ (por quê $|V| - 1$?)

   $\Rightarrow$ (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k) estarão relaxadas, **nessa ordem**.

Ideia do algoritmo de Bellman-Ford

Relaxamento de caminho: Para **todo** caminho mínimo $(v_0, v_1, \ldots, v_k)$, vamos relaxar (v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k), **nessa ordem**.

1. Se executamos **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1) estaria relaxado.
2. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2) estariam relaxados, **nessa ordem**.
3. Se executamos novamente **Relax** para todos os arcos:
   $\Rightarrow$ (v_0, v_1), (v_1, v_2), (v_2, v_3) estariam relaxados, **nessa ordem**.

   Repetimos $|V| - 1$ vezes ... (por quê $|V| - 1$?)

   $\Rightarrow$ (v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k) estarão relaxadas, **nessa ordem**.

Verificar se há ciclos negativos: repetir $+1$ vez, se um $d[v]$ mudou $\Rightarrow$ há ciclo negativo.

O algoritmo de Bellman-Ford

Entrada:

- Grafo direcionado G, função de peso nos arcos, origem s.

Saída:

- **FALSE** se existe um ciclo negativo atingível a partir de s, ou
- **TRUE** caso contrário. Neste caso, também devolve:
  vetor $d[v]$ de distâncias, vetor π (uma árvore de caminhos mínimos).

Algoritmo de Bellman-Ford

BellmanFord(G, w, s)

```
InitializeSingleSource(G, s)
para cada i ∈ [1, 2, ..., |V| − 1] faça
    para cada (u, v) ∈ E faça
        Relax(u, v, w)
para cada (u, v) ∈ E faça
    se d[v] > d[u] + w(u, v) então devolva FALSE
devolva TRUE, d, π
```

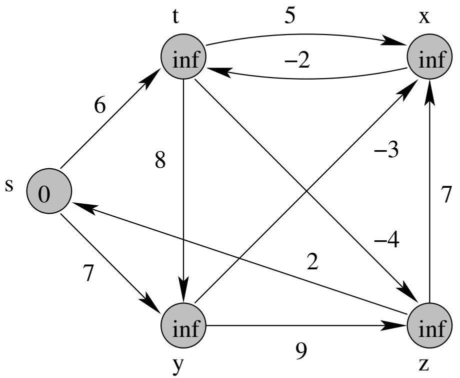
Algoritmo de Bellman-Ford

BellmanFord(G, w, s)

InitializeSingleSource(G, s)
para cada $i \in [1, 2, \ldots, |V| - 1]$ **faça**
    para cada $(u, v) \in E$ **faça**
        Relax(u, v, w)
para cada $(u, v) \in E$ **faça**
    se $d[v] > d[u] + w(u, v)$ **então devolva** FALSE
devolva TRUE, d, π

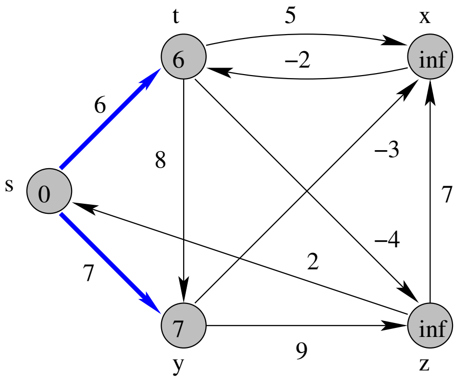
Complexidade de tempo: $O(V \cdot E)$.

Algoritmo de Bellman-Ford: exemplo



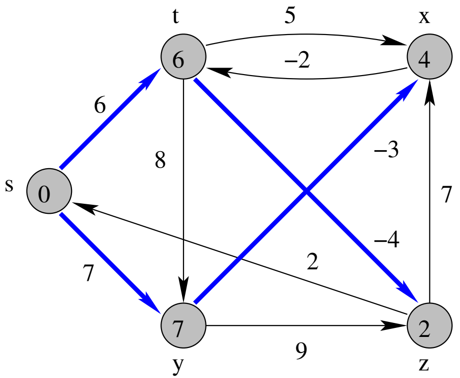
Ordem dos arcos: $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$.

Algoritmo de Bellman-Ford: exemplo



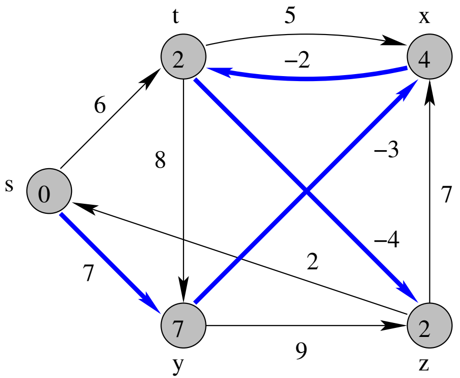
Ordem dos arcos: $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$.

Algoritmo de Bellman-Ford: exemplo



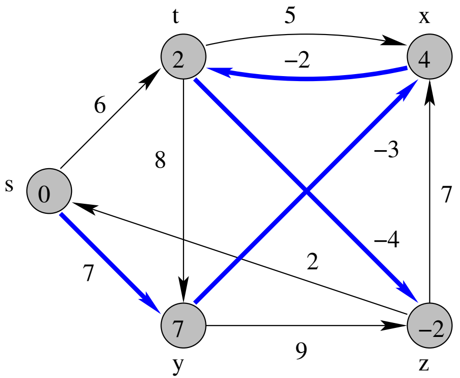
Ordem dos arcos: $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$.

Algoritmo de Bellman-Ford: exemplo



Ordem dos arcos: $(t,x),(t,y),(t,z),(x,t),(y,x),(y,z),(z,x),(z,s),(s,t),(s,y)$.

Algoritmo de Bellman-Ford: exemplo



Ordem dos arcos: $(t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)$.

Correção do algoritmo Bellman-Ford

Correção do algoritmo:

BellmanFord devolve:

- FALSE se existe um ciclo negativo atingível a partir de s,
- TRUE caso contrário. Neste caso devolve também:
  - Vetor d com $d[v] = \text{dist}(s, v)$ para todo $v \in V$.
  - Vetor π definindo uma árvore de caminhos mínimos.

Correção do algoritmo Bellman-Ford

Caso 1: Vamos supor que **não há ciclos negativos** atingíveis por s.

Correção do algoritmo Bellman-Ford

Caso 1: Vamos supor que **não há ciclos negativos** atingíveis por s.

Considere um vértice $v \in V$, e os valores de d e π após o primeiro laço:

- Se v não é atingível, $d[v] = \infty$ por **Inexistência de caminho**.
- Senão, existe caminho mínimo $(v_0, v_1, \ldots, v_k)$ de $s = v_0$ a $v = v_k$.

Correção do algoritmo Bellman-Ford

Caso 1: Vamos supor que **não há ciclos negativos** atingíveis por s.

Considere um vértice $v \in V$, e os valores de d e π após o primeiro laço:

- Se v não é atingível, $d[v] = \infty$ por **Inexistência de caminho**.
- Senão, existe caminho mínimo $(v_0, v_1, \ldots, v_k)$ de $s = v_0$ a $v = v_k$.
- Como o número de arcos no caminho $k \leq |V| - 1$, $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ foram relaxados nessa ordem.

Correção do algoritmo Bellman-Ford

Caso 1: Vamos supor que **não há ciclos negativos** atingíveis por s.

Considere um vértice $v \in V$, e os valores de d e π após o primeiro laço:

- Se v não é atingível, $d[v] = \infty$ por **Inexistência de caminho**.
- Senão, existe caminho mínimo $(v_0, v_1, \ldots, v_k)$ de $s = v_0$ a $v = v_k$.
- Como o número de arcos no caminho $k \leq |V| - 1$, $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ foram relaxados nessa ordem.
- Pela prop. de **Relaxamento do caminho**, $d[v] = \text{dist}(s, v)$.

Correção do algoritmo Bellman-Ford

Caso 1: Vamos supor que **não há ciclos negativos** atingíveis por s.

Considere um vértice $v \in V$, e os valores de d e π após o primeiro laço:

- Se v não é atingível, $d[v] = \infty$ por **Inexistência de caminho**.
- Senão, existe caminho mínimo $(v_0, v_1, \ldots, v_k)$ de $s = v_0$ a $v = v_k$.
- Como o número de arcos no caminho $k \leq |V| - 1$, $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ foram relaxados nessa ordem.
- Pela prop. de **Relaxamento do caminho**, $d[v] = \text{dist}(s, v)$.
- Pela prop. do **Sugrafo de predecessores**, π induz um caminho mínimo de s a v.

Correção do algoritmo Bellman-Ford

Caso 1: Vamos supor que **não há ciclos negativos** atingíveis por s.

Considere um vértice $v \in V$, e os valores de d e π após o primeiro laço:

- Se v não é atingível, $d[v] = \infty$ por **Inexistência de caminho**.
- Senão, existe caminho mínimo $(v_0, v_1, \ldots, v_k)$ de $s = v_0$ a $v = v_k$.
- Como o número de arcos no caminho $k \leq |V| - 1$, $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ foram relaxados nessa ordem.
- Pela prop. de **Relaxamento do caminho**, $d[v] = \text{dist}(s, v)$.
- Pela prop. do **Sugrafo de predecessores**, π induz um caminho mínimo de s a v.
- Nesse caso, **BellmanFord** devolve TRUE:
  - $d[v]$ alcançou o valor $\text{dist}(s, v)$ após o primeiro laço, pela prop. de **Convergência** ele nunca mais muda, os testes da linha 6 falham, e o algoritmo devolve TRUE.

Correção do algoritmo Bellman-Ford

Caso 2: Vamos supor agora que **existe um ciclo negativo** alcançável por s. Queremos mostrar que o algoritmo devolve FALSE.

Correção do algoritmo Bellman-Ford

Caso 2: Vamos supor agora que **existe um ciclo negativo** alcançável por s.

Queremos mostrar que o algoritmo devolve FALSE.

- Seja $C = (v_0, v_1, \ldots, v_k = v_0)$ um ciclo tal que

$$w(C) = \sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$$

- Suponha por contradição que o algoritmo devolve TRUE.

Correção do algoritmo Bellman-Ford

Caso 2: Vamos supor agora que **existe um ciclo negativo** alcançável por s.

Queremos mostrar que o algoritmo devolve FALSE.

- Seja $C = (v_0, v_1, \ldots, v_k = v_0)$ um ciclo tal que

$$w(C) = \sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$$

- Suponha por contradição que o algoritmo devolve TRUE.
- Como relaxamos cada arco (v_{i-1}, v_i),

$$d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$$

Correção do algoritmo Bellman-Ford

Somando as desigualdades anteriores para cada arco do ciclo:

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} (d[v_{i-1}] + w(v_{i-1}, v_i))$$

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} d[v_{i-1}] + \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Correção do algoritmo Bellman-Ford

Somando as desigualdades anteriores para cada arco do ciclo:

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} (d[v_{i-1}] + w(v_{i-1}, v_i))$$
$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} d[v_{i-1}] + \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Como temos um ciclo ($v_0 = v_k$), $\sum_{i=1}^{k} d[v_i] = \sum_{i=1}^{k} d[v_{i-1}]$.

Correção do algoritmo Bellman-Ford

Somando as desigualdades anteriores para cada arco do ciclo:

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} (d[v_{i-1}] + w(v_{i-1}, v_i))$$

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} d[v_{i-1}] + \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Como temos um ciclo ($v_0 = v_k$), $\sum_{i=1}^{k} d[v_i] = \sum_{i=1}^{k} d[v_{i-1}]$. Então:

$$0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C).$$

Correção do algoritmo Bellman-Ford

Somando as desigualdades anteriores para cada arco do ciclo:

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} (d[v_{i-1}] + w(v_{i-1}, v_i))$$

$$\sum_{i=1}^{k} d[v_i] \leq \sum_{i=1}^{k} d[v_{i-1}] + \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

Como temos um ciclo ($v_0 = v_k$), $\sum_{i=1}^{k} d[v_i] = \sum_{i=1}^{k} d[v_{i-1}]$. Então:

$$0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C).$$

Contradição (pois C é um ciclo negativo). O algoritmo devolve FALSE.

Problema

Problema. Uma empresa industrial quer produzir um produto X em larga escala usando uma linha de montagem. Um conjunto de etapas precisam ser executadas para produzir cada produto, e o empresario quer organizar a execução de todas as etapas do processo de produção. Porém, ele sabe que diversas etapas possuem restrições de tempo. Por exemplo, a etapa 5 precisa ser executada 2 horas depois da etapa 4; a etapa 8 precisa ser executada 24 horas depois da etapa 2.

Problema

Problema. Uma empresa industrial quer produzir um produto X em larga escala usando uma linha de montagem. Um conjunto de etapas precisam ser executadas para produzir cada produto, e o empresario quer organizar a execução de todas as etapas do processo de produção. Porém, ele sabe que diversas etapas possuem restrições de tempo. Por exemplo, a etapa 5 precisa ser executada 2 horas depois da etapa 4; a etapa 8 precisa ser executada 24 horas depois da etapa 2.

Variantes. Eventos durante um congresso: você precisa X minutos para imprimir os diplomas, os avaliadores precisam Y minutos para votar os melhores trabalhos.

Problema

Problema. Uma empresa industrial quer produzir um produto X em larga escala usando uma linha de montagem. Um conjunto de etapas precisam ser executadas para produzir cada produto, e o empresario quer organizar a execução de todas as etapas do processo de produção. Porém, ele sabe que diversas etapas possuem restrições de tempo. Por exemplo, a etapa 5 precisa ser executada 2 horas depois da etapa 4; a etapa 8 precisa ser executada 24 horas depois da etapa 2.

Variantes. Eventos durante um congresso: você precisa X minutos para imprimir os diplomas, os avaliadores precisam Y minutos para votar os melhores trabalhos.

Vamos **modelar o problema** usando variáveis e desigualdades:

- x_i representa o momento de tempo da etapa i;
- Etapa j precisa ser executada b horas depois da etapa i:

$$x_j \geq x_i + 2, \text{ ou } x_j - x_i \geq 2, \text{ ou } x_i - x_j \leq -2.$$

Sistemas de restrições de diferença

$$
\begin{array}{rcr}
x_1 - x_2 & \leq & 0 \\
x_1 - x_5 & \leq & -1 \\
x_2 - x_5 & \leq & 1 \\
x_3 - x_1 & \leq & 5 \\
x_4 - x_1 & \leq & 4 \\
x_4 - x_3 & \leq & -1 \\
x_5 - x_3 & \leq & -3 \\
x_5 - x_4 & \leq & -3
\end{array}
$$

Sistemas de restrições de diferença

$$\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}$$

Vamos usar algoritmos para o problema de **caminhos mínimos** para resolver o sistema, i.e., é encontrar $x_1, x_2, \ldots, x_n$ que satisfaçam todas as desigualdades.

Sistemas de restrições de diferença

Podemos reescrever as restrições de forma matricial:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

Sistemas de restrições de diferença

Podemos reescrever as restrições de forma matricial:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

Algumas soluções: $x = (-5, -3, 0, -1, -4)$,

Sistemas de restrições de diferença

Podemos reescrever as restrições de forma matricial:

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \\ 1 \\ 5 \\ 4 \\ -1 \\ -3 \\ -3 \end{pmatrix}$$

Algumas soluções: $x = (-5, -3, 0, -1, -4)$, $x' = (0, 2, 5, 4, 1)$, ...

Sistemas de restrições de diferença

Propriedade

Seja $x = (x_1, \ldots, x_n)$ uma solução de um sistema de restrições de diferença $Ax \leq b$ e d uma constante. Então

$$x + d = (x_1 + d, \ldots, x_n + d)$$

também é uma solução de $Ax \leq b$.

Sistemas de restrições de diferença

Propriedade

Seja $x = (x_1, \ldots, x_n)$ uma solução de um sistema de restrições de diferença $Ax \leq b$ e d uma constante. Então

$$x + d = (x_1 + d, \ldots, x_n + d)$$

também é uma solução de $Ax \leq b$.

Veja que $(x_j + d) - (x_i + d) = x_j - x_i$, para cada par x_i, x_j em cada desigualdade. ■

Sistemas de restrições de diferença

Propriedade

Seja $x = (x_1, \ldots, x_n)$ uma solução de um sistema de restrições de diferença $Ax \leq b$ e d uma constante. Então

$$x + d = (x_1 + d, \ldots, x_n + d)$$

também é uma solução de $Ax \leq b$.

Veja que $(x_j + d) - (x_i + d) = x_j - x_i$, para cada par x_i, x_j em cada desigualdade. ■

Mostraremos a seguir como encontrar uma solução de um sistema $Ax \leq b$ de restrições de diferença resolvendo o problema de **caminhos mínimos**.

Grafo de restrições

Construímos o chamado **grafo de restrições**:

1. Primeiro criamos um grafo em que:
    - Cada vértice v_i corresponde a uma variável x_i;
    - Cada arco (v_i, v_j) com custo b_k corresponde a uma restrição $x_j - x_i \leq b_k$.

Grafo de restrições

Construímos o chamado **grafo de restrições**:

1. Primeiro criamos um grafo em que:
   - Cada vértice v_i corresponde a uma variável x_i;
   - Cada arco (v_i, v_j) com custo b_k corresponde a uma restrição $x_j - x_i \leq b_k$.
2. Agora adicionamos um vértice especial v_0 e uma aresta de v_0 a cada outro vértice v_i com custo 0.

Grafo de restrições

Formalmente, dado o sistema $Ax \leq b$ de restrições de diferença, construímos o grafo direcionado $G = (V, E)$ tal que

- $V = \{v_0, v_1, \ldots, v_n\}$;
- $E = \{(v_i, v_j) : x_j - x_i \leq b_k$ é restrição$\} \cup \{(v_0, v_1), \ldots, (v_0, v_n)\}$.

Grafo de restrições

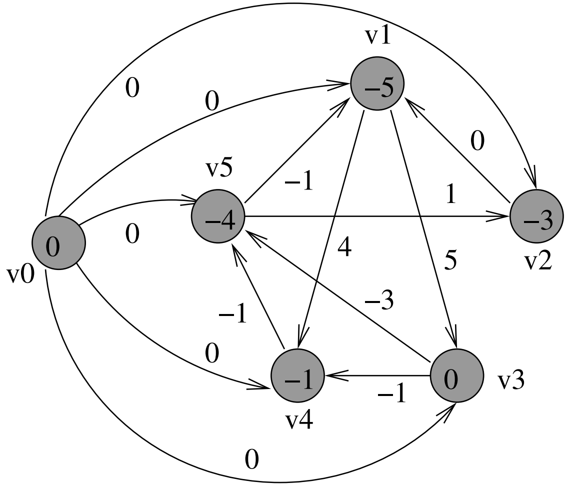
Formalmente, dado o sistema $Ax \leq b$ de restrições de diferença, construímos o grafo direcionado $G = (V, E)$ tal que

- $V = \{v_0, v_1, \ldots, v_n\}$;
- $E = \{(v_i, v_j) : x_j - x_i \leq b_k \text{ é restrição}\} \cup \{(v_0, v_1), \ldots, (v_0, v_n)\}$.

E associamos os custos:

- $w(v_i, v_j) = \begin{cases} b_k & \text{se } x_j - x_i \leq b_k \text{ é restrição} \\ 0 & \text{se } i = 0 \end{cases}$

Grafo de restrições



Grafo de restrições

Seja $Ax \leq b$ um sistema de restrições de diferença e $G = (V, E)$ o grafo de restrições associado a esse sistema. Então:

- Se G não contém ciclos negativos, então

$$x = (\text{dist}(v_0, v_1), \text{dist}(v_0, v_2), \dots, \text{dist}(v_0, v_n))$$

  é uma solução viável do sistema;
- Se G contém ciclos negativos, então o sistema não possui solução viável.

Resolvendo um sistema de restrições de diferença

Para **resolver o sistema** de restrições de diferença, basta resolver caminhos mínimos:

- Executar **BellmanFord** a partir de v_0 no grafo de restrições G.
- Todo vértice é alcançável de v_0: se há ciclo negativo, o algoritmo devolve FALSE.
- Se não há ciclo negativo, então a solução é $x = (d[v_1], d[v_2], \ldots, d[v_n])$.

Resolvendo um sistema de restrições de diferença

Para **resolver o sistema** de restrições de diferença, basta resolver caminhos mínimos:

- Executar **BellmanFord** a partir de v_0 no grafo de restrições G.
- Todo vértice é alcançável de v_0: se há ciclo negativo, o algoritmo devolve FALSE.
- Se não há ciclo negativo, então a solução é $x = (d[v_1], d[v_2], \ldots, d[v_n])$.

Tempo de execução do algoritmo:

- A matriz A tem dimensões $m \times n$.

Resolvendo um sistema de restrições de diferença

Para **resolver o sistema** de restrições de diferença, basta resolver caminhos mínimos:

- Executar **BellmanFord** a partir de v_0 no grafo de restrições G.
- Todo vértice é alcançável de v_0: se há ciclo negativo, o algoritmo devolve FALSE.
- Se não há ciclo negativo, então a solução é $x = (d[v_1], d[v_2], \ldots, d[v_n])$.

Tempo de execução do algoritmo:

- A matriz A tem dimensões $m \times n$.
- G possui $n + 1$ vértices e $n + m$ arestas.

Resolvendo um sistema de restrições de diferença

Para **resolver o sistema** de restrições de diferença, basta resolver caminhos mínimos:

- Executar **BellmanFord** a partir de v_0 no grafo de restrições G.
- Todo vértice é alcançável de v_0: se há ciclo negativo, o algoritmo devolve FALSE.
- Se não há ciclo negativo, então a solução é $x = (d[v_1], d[v_2], \ldots, d[v_n])$.

Tempo de execução do algoritmo:

- A matriz A tem dimensões $m \times n$.
- G possui $n + 1$ vértices e $n + m$ arestas.
- O tempo de execução de **BellmanFord** em G é $O((n+1)(n+m)) = O(n^2 + nm)$.

Resumo

Síntese

- O algoritmo de **Bellman-Ford** resolve o problema de encontrar os **caminhos mínimos de fonte única**, mesmo quando há arcos de peso negativo (mas não há ciclos negativos).
- O algoritmo de **Bellman-Ford** consegue identificar quando há ciclo negativo alcançável a partir do vértice inicial, e nesse caso devolve **FALSE**.
- **Sistemas de restrições de diferença** podem ser resolvidos por meio de uma aplicação direta do algoritmo de **Bellman-Ford** no grafo de restrições.

Material bibliográfico e exercícios

T. Cormen et al. Algoritmos - Teoria e Prática (3a ed.). – **Cap. 24**

Exercícios: ver exercícios no final dos (sub)capítulos do Cap. 24.

Dúvidas

Dúvidas?