TÉCNICO LISBOA

Análise e Síntese de Algoritmos

SSSPs: Johnson

CLRS Cap. 25

Instituto Superior Técnico

2022/2023

Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Caminhos mais curtos [CLRS, Cap.22,24-25]
  - Árvores abrangentes
  - Fluxos máximos
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Complexidade Computacional

Resumo

Algoritmo Johnson

Algoritmo Johnson

Intuição

- Utiliza algoritmo de Dijkstra para cada vértice
- Efetua repesagem dos arcos, para eliminar pesos negativos: calcula novo conjunto de pesos não negativos w', tal que:
  - Um caminho mais curto de u para v com função w é também caminho mais curto com função w'
  - Para cada arco (u, v) o peso $w'(u, v)$ é não negativo
- Utiliza algoritmo de Bellman-Ford no procedimento de repesagem e para detetar ciclos negativos

Repesagem Não Altera Caminhos Mais Curtos

- Dado $G = (V, E)$, com função de pesos w e de repesagem $h : V \to \mathbb{R}$, para cada arco $(u, v) \in E$ temos:

$$w'(u, v) = w(u, v) + h(u) - h(v)$$

- Seja $p = \langle v_0, v_1, \ldots, v_k \rangle$ um caminho de v_0 para v_k. Então, p é um caminho mais curto de v_0 para v_k utilizando w se e só se é um caminho mais curto utilizando w'
  - $w(p) = \delta(v_0, v_k)$ sse $w'(p) = \delta'(v_0, v_k)$
  - $w'(p) = \delta'(v_0, v_k) = \delta(v_0, v_k) + h(v_0) - h(v_k)$

Repesagem Não Altera Caminhos Mais Curtos (cont.)

Prova

- Verificar que $w'(p) = w(p) + h(v_0) - h(v_k)$

$$\begin{aligned} w'(p) &= \sum_{i=1}^{k} w'(v_{i-1}, v_i) \\ &= \sum_{i=1}^{k} (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) \\ &= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k) \\ &= w(p) + h(v_0) - h(v_k) \end{aligned}$$

Repesagem Não Altera Caminhos Mais Curtos (cont.)

Prova

- Para qualquer caminho p de v_0 para v_k, temos:

$$w'(p) = w(p) + h(v_0) - h(v_k)$$

- Como $h(v_0)$ e $h(v_k)$ não dependem do caminho, se tivermos dois caminhos p_1 e p_2 tal que $w(p_1) < w(p_2)$, então $w'(p_1) < w'(p_2)$:

$$\begin{gathered} w(p_1) < w(p_2) \\ w'(p_1) - h(v_0) - h(v_k) < w'(p_2) - h(v_0) - h(v_k) \\ w'(p_1) < w'(p_2) \end{gathered}$$

- Logo, $w(p) = \delta(v_0, v_k)$ se e só se $w'(p) = \delta'(v_0, v_k)$

Ciclos Negativos

- Existe ciclo negativo utilizando w se e só se existe ciclo negativo utilizando w'

Prova

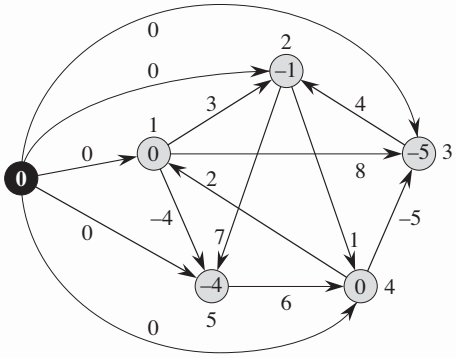
- Considerar ciclo $c = \langle v_0, v_1, \ldots, v_k \rangle$, onde $v_0 = v_k$
- Atendendo ao que foi derivado anteriormente, e observando que $h(v_0) = h(v_k)$ (dado que $v_0 = v_k$), obtemos:

$$\begin{aligned} w'(c) &= w(c) + h(v_0) - h(v_k) \\ &= w(c) \end{aligned}$$

- Logo, $w(c) < 0$ se e só se $w'(c) < 0$

Organização

- Dado $G = (V, E)$, criar $G' = (V', E')$ tal que:
  - $V' = V \cup \{s\}$
  - $E' = E \cup \{(s, v) : v \in V\}$
  - $\forall v \in V : w(s, v) = 0$



Organização

- Ciclos negativos são detectados pela execução do algoritmo de Bellman-Ford aplicado a G'
- Se não existirem ciclos negativos:
  - Definir $h(v) = \delta(s, v)$
  - Pela propriedade dos caminhos mais curtos, para cada arco (u, v), temos que $h(v) \leq h(u) + w(u, v)$
  - Logo, $w'(u, v) = w(u, v) + h(u) - h(v) \geq 0$
- Executar Dijkstra para todo o $u \in V$ com função de peso w'
  - Cálculo $\delta'(u, v)$ para todo o $u \in V$
  - Mas também $\delta(u, v) = \delta'(u, v) - h(u) + h(v)$

Johnson(G)
  representar G'
  if Bellman-Ford$(G', w, s) == FALSE$ **then**
    return "Indicar ciclo negativo"
  else
    $h(v) = \delta(s, v)$, calculado com Bellman-Ford
    for each $(u, v) \in G.E$ **do**
      $w'(u, v) = w(u, v) + h(u) - h(v)$
    end for
    for each $u \in G.V$ **do**
      executar Dijkstra(G, w', u)
      calcular $\delta'(u, v)$
      for each $v \in G.V$ **do**
        $d_{uv} = \delta'(u, v) + h(v) - h(u)$
      end for
    end for
  end if
  return D

Complexidade

- Bellman-Ford: $O(VE)$
- Executar Dijkstra para cada vértice: $O(V(V + E)\ lg\ V)$ (assumindo amontoado binário)
- Total: $O(V(V + E)\ lg\ V)$
- Útil para grafos esparsos