10 / 05 / 24

Problema do Caminho Mínimo

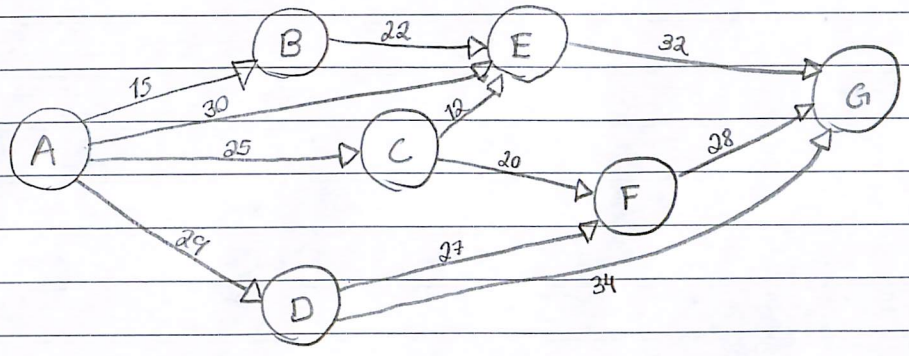
o problema do caminho mínimo consiste em encontrar o melhor caminho entre dois nós. Resolver esse problema pode significar determinar o caminho entre dois nós com menor custo / tempo de viagem.
↳ problema clássico que utiliza representação por grafos.

numa rede qualquer, podem existir vários caminhos entre um par de nós, definidos como origem e destino. entre os vários caminhos, o que possui menor "peso" é chamado de caminho mínimo. este peso representa a soma total dos valores dos arcos/arestas que compõe o caminho. estes valores podem representar o tempo da viagem, a distância percorrida, etc.

▽ esses problemas podem ser resolvidos por meio do método Simplex, como vimos. porém, existem métodos + eficientes como o algoritmo de Dijkstra ou o de Bellman-Ford

representação por grafos

- vértices representam cidades
- arestas/arcos representam estradas ou caminhos
  ↳ custo de cada aresta = distância ou tempo
  ↳ no simplex, custo = C_{ij} !



/ /

construção do modelo simplex

custos: C_{ij} é o custo de cada aresta

variáveis de decisão: X_{ij} representa a passagem (ou não) pela aresta / caminho de i para j.

↳ X é booleano! pode assumir 1 = passagem ou 0 = não passagem.

função objetivo: queremos minimizar a distância entre a origem e destino, percorrendo os caminhos possíveis.

↳ $Z = \sum_{i=1}^{n} \sum_{j \in S(i)} C_{ij} \cdot X_{ij}$ (min) → S(i) = conj. nós sucessores de i

↳ $\min Z = C_{AB} \cdot X_{AB} + \ldots + C_{ij} X_{ij}$

(soma de todos os caminhos possíveis multiplicados pelo seu custo).

↳ obs (se aresta ↔, precisamos considerar $X_{AB} \neq X_{BA}$. se →, só existe X_{AB}!!)

restrições: restrições são chamadas de restrições de fluxo, e são feitas para cada nó (cidade). elas representam a DIFERENÇA ENTRE A CHEGADA E SAÍDA DE CADA NÓ (quanto chega − quanto sai). → chegadas / saídas = X_{ij}

↳ nó N = chegadas − saídas

- para a origem, a diferença = -1 (pois só saímos!)
- para o destino, a diferença = 1 (pois só chegamos)
- para nós intermediários, a dif. = 0 (pois, se chegamos, precisamos sair!)

↳ como sempre, $X_{ij} \geq 0$

ex.

origem: $-X_{AB} - X_{AC} - X_{AD} - X_{AE} = -1$

nó n: $X_{AD} - X_{DF} - X_{DG} = 0$

destino: $X_{DG} + X_{FG} + X_{FG} = 1$

11 / 05 / 24

Algoritmo de Dijkstra

solução para o problema do caminho mínimo aceitando custos positivos

representação

1	2*	...	N
0	∞	∞	∞
—	⋮	⋮	⋮
—	—	—	(V_A, C_T)

- 2*: indica se vértice já é ou não rotulado
- N: número de vértices
- V_A = vértice anterior, C_T = custo total
- —: vértice já foi rotulado e não é + considerado

passo-a-passo

- o algoritmo define um nó ls como rotulado quando se encontra no menor caminho, e não-rotulado quando o caminho mínimo AINDA não foi encontrado.
- ao início, rotulados = ∅ e não-rotulados = {1, 2, ..., N}

1. atribuir 0 ao nó origem e ∞ aos demais
   - valores representam os custos de deslocamento

2. enquanto o conjunto dos não-rotulados não for vazio, faça:

3. selecione o nó não-rotulado com menor valor de deslocamento (C_T) = K
   - na 1ª iteração, será o nó origem. em todas as outras, será a célula com menor C_T

4. passe o nó selecionado (K) para o conjunto dos rotulados
   - indique com * na coluna e selecione a célula (V_A, C_T)

5. passe para a próxima linha da tabela e preencha cada coluna:
   - se o nó/coluna já foi rotulado, célula = —
   - se o nó NÃO é sucessor de K, repetir o valor da linha anterior
   - se o nó É sucessor de K, calcular o novo C_T
     - $\overline{C_T} = C_T$ anterior + C_{ij} (custo para chegar de K ao sucessor)
     - se $\overline{C_T}$ for MENOR do que o C_T anterior da coluna, o novo valor é atualizado → $(V_A = K, C_T = \overline{C_T})$. se o valor C_T da linha anterior é menor, não atualizamos o valor, apenas copiamos a linha anterior.

análise dos resultados

ao relaxarmos todos os nós, o algoritmo termina.
↳ o último (V_A, C_T) na coluna do nó DESTINO indica:
- C_T = custo total do caminho mínimo
- V_A = o vértice/nó anterior do destino

ou seja, precisamos percorrer o caminho de trás para frente para obter o caminho mínimo!

começando da coluna/nó DESTINO, analisar o último V_A da coluna
↳ se $V_A = X$, ir à coluna/nó X e analisar o último V_A até chegarmos no nó de origem!

Problema do Caminho Mínimo

problema do caminho mínimo busca encontrar o menor caminho entre dois nós de uma rede. podemos minimizar distância, custo e/ou tempo!

↳ problema considera apenas um nó de oferta (ponto de origem) e apenas um nó de demanda (ponto de destino). a capacidade de fornecimento do nó de oferta e demanda do nó de destino correspondem a UMA unidade. todos os outros nós intermediários tem oferta/demanda ZERO.

parâmetros

C_{ij} = distância de i pl j.

vars de decisão

$X_{ij} = 1$ ou 0 (1 se percorrem pela aresta, 0 se não)

função objetivo: $C_{12}X_{12} + \ldots + C_{ij} \cdot X_{ij}$

↳ TODOS os caminhos possíveis (custo x X (se percorrem ou não))

algoritmo de Dijkstra: determina o menor caminho assumindo custos não-negativos. algoritmo define um nó K como rotulado ou fechado quando se encontra o menor caminho. já os nós cujos caminhos mínimos ainda não foram encontrados são não-rotulados, ou abertos.

↳ R = conj. nós rotulados. NR = conj. não rotulados

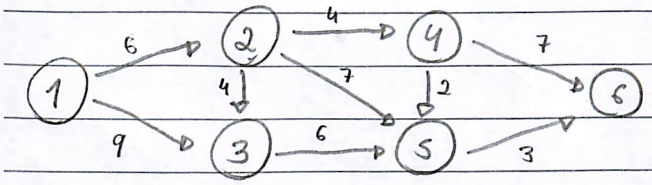
passo a passo.

início: $R = \emptyset$, $NR = \{1, 2, \ldots, n\}$

1. atribua 0 ao nó fonte e ∞ aos demais
2. enquanto NR não for vazio, faça: ↳ C_{ij}
   - selecione o nó não-rotulado com menor valor (K)
   - passe no K para conjunto dos rotulados
   - para todo nó J sucessor de K ("distinto" direto de K), faça:
     - some o valor de K c/ o valor do arco que une K e J e atribua a J, se houver melhoria. se houver, definimos K como predecessor de J.

/ /

ex. 2



R =

NR =

solução

1* (rotulado)	2*	3*	4*	5*	6*
0	∞	∞	∞	∞	∞
—	**(1,6)**	(1,9)	∞	∞	∞
—	—	**(1,9)**	(2,10)	(2,13)	∞
—	—	—	**(2,10)**	(2,13)	∞
—	—	—	—	**(4,12)**	(4,17)
—	—	—	—	—	**(5,15)**

1. ① = menor valor, rotular

② ② e ③ não vêm de ①, fazemos $(1, 1+c_{12})$ e $(1, 1+c_{13})$
↳ (1,6) é o menor caminho!

③ rotulamos o ②

④ $(2, 6+c_{23}) = (2, 6+4) = (2, 10)$
↳ não atualizamos (1,9) pois 9<10!! (já atual; não atualizamos na linha mesmo)

$(2, 6+c_{25}) = (2, 6+7) = (2, 13)$

$(2, 6+c_{24}) = (2, 6+4) = (2, 10)$

vindo do 2, menor custo é pro ③

⑤ rotular o ③

⑥ calcular vizinhos do 3

$(3, 9+c_{35}) = (3, 9+6) = (3, 15)$

15 > 13 (custo anterior do 5), mantém

⑦ calcular vizinhos do 4

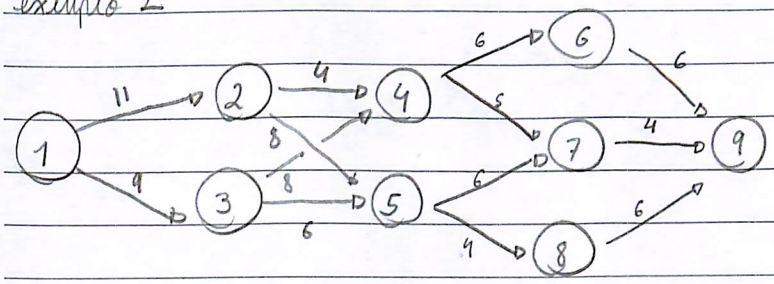
5: (4, 12), 6: (4, 17)

⑧ calcular vizinhos do 5

$(5, 12+c_{56}) = (5, 12+3) = (5, 15)$

↳ logo, menor caminho é 1 → 2 → 3 → 4 → 5 → 6
com custo = 15!

exemplo 1



obs.
$X_{12} = 1 \rightarrow 2$ //
$X_{12} = 1$ se está no caminho + curto.
0 se não.

1*	2*	3*	4*	5*	6*	7*	8*	9*	
0	∞	∞	∞	∞	∞	∞	∞	∞	start em 1, 1 rotulado
—	(1,11)	[(1,9)]	∞	∞	∞	∞	∞	∞	rotulamos o 3
—	[(1,11)]	—	(3,17)	(3,15)	∞	∞	∞	∞	repito o 2, calculo sucessores do 3. rotulo 2
—	—	—	[(2,15)]	(3,15)	∞	∞	∞	∞	calculo sucessores do 2. repito 5 e atualizo 4 ↳ e rotulo !!
—	—	—	—	[(3,15)]	(4,21)	(4,20)	∞	∞	repito 5, calculo sucessores do 4, rotulo 5
—	—	—	—	—	(4,21)	(4,20)	[(5,19)]	∞	repito 6, calculo suc. de 5 (7 e 8). rotulo 8.
—	—	—	—	—	(4,21)	[(4,20)]	—	(8,25)	repito 6 e 7, calculo suc. de 8 (9). rotulo 7
—	—	—	—	—	[(4,21)]	—	—	(7,24)	repito 6, atualizo 9, rotulo 6
—	—	—	—	—	—	—	—	[(7,24)]	rotulo 9.
—	—	—	—	—	—	—	—	—	↳ custo 24

9
7
4
2
1

1 → 2 → 4 → 7 → 9

obs. a cada linha, repito os valores que não podem ser melhorados, seja pq o nó não é sucessor do atual ou pq, se atingido pelo atual, o custo é maior !!

↳ partindo do último rotulado, calculo custo atual total + aresta que conecta último rotulado até cada sucessor. se o custo total é menor do que anterior, atualizo o valor. se não, mantenho o anterior.

↳ prox linha, relaciono o menor custo total e repito.

↳ quando um número entra nos rotulados, paro de considerar ele !!

obs: tudo é menor que ∞

pl determinar o caminho, parto do último e analiso a origem de cada iteração // → analisar como chegou em cada coluna (coluna rotulada em cada coluna).