Minimum Routing Cost Tree

Metodi di Ottimizzazione - AA 2021/22

Fabio Bedeschi

Problema

Dato un grafo non orientato e pesato, trovare l'albero di copertura che minimizza la somma del costo dei cammini tra ogni coppia di nodi.

Dati forniti in ingresso:

- **N**: numero di nodi presenti nel grafo (identificati con *i : 1..N*)
- **GRAPH**: matrice *NxN* con elemento *i,j=1* se il grafo presenta una connessione tra i nodi *i e j*, altrimenti *0*
- **COSTS**: matrice *NxN* con elemento *i,j*=*c* indicante il costo dell'arco *(i,j)*

Variabili del modello

- **x**: *array NxN di binary mpvar* indicanti la scelta del ramo i,j nell'albero x(i,j)
- **y**: *array NxNxM di binary mpvar* indicanti la scelta del k-esimo path fra i nodi i e j
- **PATHS**: *array NxNxM* in cui l'el. (i,j,k) contiene il k-esimo cammino di nodi per raggiungere *j* partendo dal nodo *i*
- **DISTS**: *array NxNxM* che contiene il costo del path (i,j,k)

***M** valorizzato a runtime come massima cardinalità di cammini tra tutte le coppie (i,j) di nodi nel grafo*

Variabili di lavoro

- **paths**, **visited**: set variables globali per l’esplorazione del grafo
- **violation**: flag booleano utilizzato nella generazione di tagli
- **iter**: contatore di iterazioni nella generazione dei tagli
- **v_cnt**: contatore di violazioni
- **v_ctrs**: array per la memorizzazione dei tagli imposti

Vincoli (1) (2)

$$x_{ij} = x_{ji} \quad y_{ijk} = y_{jik}$$
$$i < j \quad \forall i, j \in N \quad \forall k \in M$$

Essendo il grafo in esame non orientato, i vincoli sopra impongono la conservazione della simmetria nella soluzione scelta.

Vincoli (3)

$$\sum_{i<j}^{N} x_{ij} = N - 1 \qquad x_{ij} \in \{0, 1\}$$

Per poter ottenere un albero dal grafo è necessario imporre la scelta di esattamente N-1 archi.

Vincoli (4)

$$\sum_{k}^{M} y_{ijk} = 1 \quad t.c.\ i < j\ \ \forall i, j \in N \quad y_{ijk} \in \{0, 1\}$$

Tra ogni coppia di nodi *(i,j)*, si vuole ammettere nella soluzione un unico cammino tra i *k* cammini possibili nel grafo.

Vincoli (5)

$$\sum_{k|(i,j)\in Paths_{sdk}}^{M} y_{sdk} \leq x_{ij}$$

$$t.c. \quad i < j, \;\; s \neq d \quad \forall i, j, s, d \in N$$

Si devono scegliere solo cammini che utilizzano archi selezionati per la soluzione.

Questo significa che, per ogni arco del grafo, la somma delle var decisionali di tutti i possibili cammini che lo attraversano deve essere minore o uguale al valore della var decisionale dell’arco stesso.

Funzione di costo

$$\sum_{i<j}^{N} \sum_{k}^{M} y_{ijk} * Dists_{ijk}$$

La funzione di costo è espressa come la somma del costo di tutti i cammini selezionati nella soluzione corrente del problema.

Algoritmo risolutivo

1. Lettura dei dati di input
2. Esplorazione del grafo con enumerazione di tutti i possibili cammini fra i nodi
3. Istanziazione delle variabili decisionali e applicazione dei vincoli iniziali (vincoli 1,2,3,4)
4. Minimizzazione della funzione di costo con rilassamento continuo
5. Ricerca di possibili violazioni, se trovate:
   a. Applicazione dei nuovi vincoli dinamici (vincolo 5 sulle violazioni individuate)
   b. Ripetere dal passo 4
6. Minimizzazione della funzione obiettivo *senza* rilassamento continuo

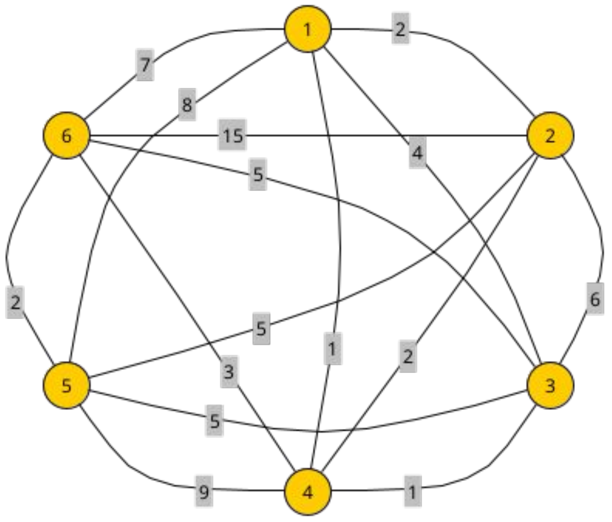
Ricerca delle violazioni

A ogni iterazione dell’algoritmo si controlla che ogni arco **non scelto** per la soluzione del problema, **non stia venendo percorso** da alcun cammino fra quelli selezionati al momento.

Se ciò non è rispettato si applica il vincolo 5 basato sulla coppia di nodi (*arco*) che ha generato la violazione e si ripete la ricerca dell’ottimo.

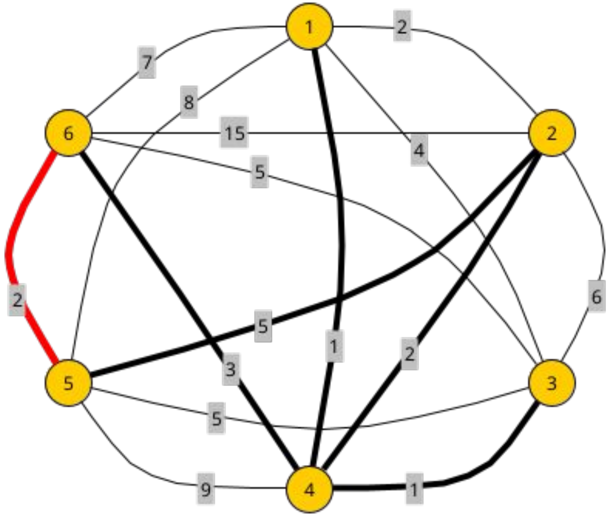
Esempio

Stato iniziale:



Esempio

Almeno un cammino scelto usa l'arco (5,6) non selezionato nella soluzione corrente:



Esempio

Soluzione: