Algoritmo che analizza solo le label potenzialmente modificate

Filippo Magi

December 23, 2021

1 Ottimizzazione sulle ultime label

Algorithm 1 Ricerca del massimo flusso a costo minimo con ricalcolo solo nelle ultime label

Require: grafo dei residui $\overleftrightarrow{G} = \{V(G), E(G) \cup \{\overleftarrow{e} : e \in E(G)\}$.
Ensure: valore del flusso massimo
  s $\leftarrow$ SourceNode di $\overleftrightarrow{G}$
  t $\leftarrow$ SinkNode di $\overleftrightarrow{G}$
  noCap $\leftarrow$ null
  loop
    f $\leftarrow$ DoBfs(grafo,noCap)
    if f = 0 **then**
      break
    end if
    mom $\leftarrow$ t
    while mom $\neq$ s **do**
      aggiorno capacità o flusso del nodo m a seconda del suo predecessore
      if capacità dell'arco (m,predecossore di m) = 0 **then**
        noCap $\leftarrow$ mom
      end if
      mom $\leftarrow$ predecessore di mom
    end while
  end loop
  return flusso uscente da s

Algorithm 2 Algoritmo DoBfs con ottimizzazione solo nelle ultime label

Require: grafo dei residui $\overleftrightarrow{G}$, nodo noCap
Ensure: valore del flusso inviato al SinkNode, $\overleftrightarrow{G}$ aggiornato
coda ← coda vuota di nodi
if noCap = null **then**
coda.Enqueue(source nodo di $\overleftrightarrow{G}$
else
rendo invalido noCap
provo a riparare noCap /*controllo se c'è un nodo con label = noCap.label-1 e con capacità o flusso diversa da 0*/
t ← SinkNode di $\overleftrightarrow{G}$
if noCap è stato riparato AND il flusso entrante nel predecessore di t $\neq 0$ AND capacità dell'arco (t, predecessore di t) > 0 **then**
return Min(flusso entrante di t, flusso passante per noCap)
end if
coda ← nodi di $\overleftrightarrow{G}$ con label = (noCap.label - 1)
cancello informazioni contenute nei nodi con label $\leq$ noCap.label
for all nodo n $\in$ coda **do**
controllo che flusso inviato di n sia legale con precedessori di n { analizzo se il predecessore abbia flusso entrate $\geq$ flusso entrante di n, in questo caso termina, altrimenti continua coi i predecessori, da valutare se deve arrivare fino a n o se basta che incontri il primo nodo con flusso legale, CorrectFlow nel codice}
end for
end if
while la coda non è vuota **do**
element ← coda.dequeue()
for all edge che entrano o escono da element **do**
n ← nodo dove edge entra
p ← nodo dove edge esce
if element è valido (cioè la label è corretta) **then**
if p = element AND capacità di edge > 0 AND (n non è stato esplorato OR n non è valido) **then**
aggiorno dati di n {label, flusso entrante e nodo precedente, nel caso sia necessario "riparo" il nodo}
if n è SinkNode di $\overleftrightarrow{G}$ **then**
return flusso entrante di n
else
coda.Enqueue(n)
end if
if n = element AND flusso di edge > 0 AND (p non è stato esplorato OR p non è valido) **then**
aggiorno i dati di p
aggiorno edge indicando che deve essere percorso in senso opposto
if p è SinkNode di $\overleftrightarrow{G}$ **then**
return flusso entrante di p
else
coda.Enqueue(p)
end if
end if
end if
end if
end for
end while
return 0