Laboratorio di Reti e Sistemi Distribuiti

18: Cammini Minimi

Roberto Marino, PhD[1]
roberto.marino@unime.it

[1]Dipartimento di Matematica, Informatica, Fisica e Scienze della Terra
Future Computing Research Laboratory
Università di Messina

Last Update: 27th May 2025

Il problema dei cammini minimi

Definizione del problema

Consideriamo una rete sincrona associata a un **digrafo pesato**, cioè assumiamo che a ogni arco di comunicazione sia associato un peso strettamente positivo. Il nostro obiettivo è calcolare un albero contenente i percorsi più brevi da una nodo sorgente, ad esempio il nodo 1, verso tutti gli altri nodi. Come per il calcolo di un albero BFS, vogliamo ottenere una rappresentazione distribuita di un albero diretto (ogni nodo computa localmente) con una memoria limitata per ogni nodo.

Algoritmo di Bellman-Ford (Dijkstra distribuito)

Descrizione Informale

Ogni agente mantiene in memoria una stima `dist` della sua distanza ponderata dalla sorgente e una stima `parent` del vicino corrispondente al percorso più breve (ponderato) dalla sorgente. La stima di `dist` è inizializzata a 0 per la sorgente e a "infinite" per tutti gli altri nodi. In ogni round di comunicazione, ogni agente esegue i seguenti compiti: (1) trasmette la stima del valore di `dist` ai suoi vicini, (2) calcola la quantità più piccola tra "il valore di distanza ricevuto da un in-neighbor sommato al peso del bordo corrispondente a quell'in-neighbor" e (3) se la stima di `dist` dell'agente è più grande di questa quantità, allora l'agente aggiorna la sua `dist` e la sua variabile `parent` stimata.

Bellman-Ford Algorithm Pseudocode

`Synchronous Network with Weights:` $\mathcal{S} = (\{1, \ldots, n\}, E_{\text{cmn}}, A)$

`Distributed Algorithm:` DISTRIBUTED BELLMAN-FORD

`Alphabet:` $\mathbb{A} = \mathbb{R}_{>0} \cup \texttt{null} \cup \{+\infty\}$

`Processor State:` $w = (\texttt{parent}, \texttt{dist})$, where

$\texttt{parent} \in \{1, \ldots, n\}$,	initially:	$\texttt{parent}^{[j]} = j$ for all j
$\texttt{dist} \in \mathbb{A}$,	initially:	$\texttt{data}^{[1]} = 0$,
		$\texttt{data}^{[j]} = +\infty$ for all $j \neq 1$

function msg(w, i)

1: **if** round $< n$ **then**
2: **return** dist
3: **else**
4: **return** null

function stf(w, y)

1: $i :=$ processor UID
2: $k := \text{arginf}\{y_j + a_{ji} \mid \text{for all } y_j \neq \texttt{null}\}$
3: **if** $(\texttt{dist} < k)$ **then**
4: **return** $(\texttt{parent}, \texttt{dist})$
5: **else**
6: **return** $(k, y_k + a_{ki})$

Bellman-Ford: Analisi di Complessità

Per un grafo di ordine n:

Complessità temporale

Uguale a $n - 1$

Complessità di comunicazione

Complessità in $\Theta(n * dim(E_{cmm}))$

Bellman-Ford Algorithm

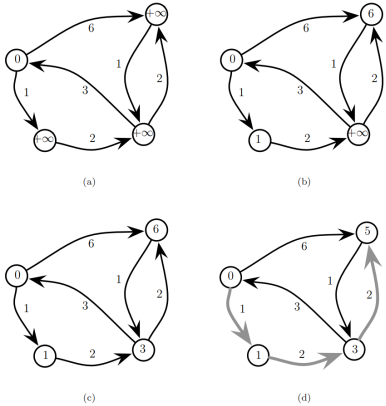

Figure 1.18 Execution of the DISTRIBUTED BELLMAN-FORD ALGORITHM. (a) The processor state initialization. The vertex 1 is the only one whose variable `dist` is 0. After three iterations, as guaranteed by Lemma 1.58, (d) depicts the resulting shortest-paths tree of the digraph rooted at vertex 1. This tree is represented in the last frame, with edges colored in gray.

Errata corrige: riga 3, funzione stf()

A riga 3 della funzione stf() bisogna fare il confronto tra la variabile `dist` ed il valore $y_k + a_{ki}$