Minimum Bottleneck Spanning Tree

Yuxiang Cui

Nov. 25 2013

Outline

- Undirected Graph

  Camerini's Algorithm 1

- Directed Graph

  Camerini's Algorithm 2

  Gabow and Tarjan's Algorithm

Background

- Bottleneck

  The largest edge in a spanning tree

- Minimum Bottleneck Spanning Tree (MBST)

  A spanning tree whose bottleneck edge weight is minimum

- MBST problem

  How to get a MBST？

Camerini's Algorithm 1

- Definition

  G = (V, E), A = a subset of E, B = E except A

  All edges' weights in A >= B

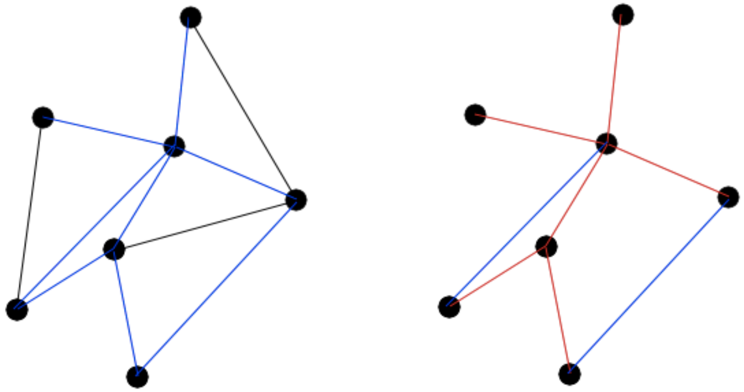
  F = a maximal forest of GB

- Theorem 1

  (a) If F is a spanning tree of G, a MBST of G is given by any MBST of GB. (b) Otherwise, it can be obtained by adding to F any MBST of G', where G' is the graph GA collapsed.

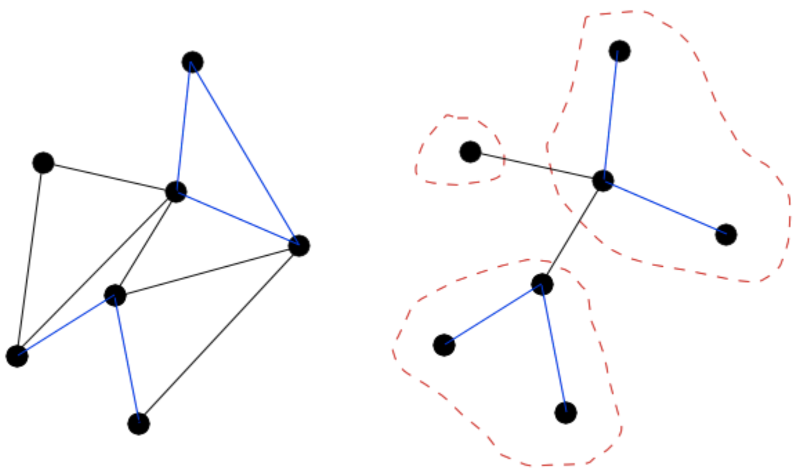
Camerini's Algorithm 1

- Theorem 1 (a)



Camerini's Algorithm 1

- Theorem 1 (b)



Camerini's Algorithm 1

- E := the set of edges of G

  if | E | = 1 then return E else

  A := UH(E, w)

  B := E - A

  F := FOREST (GB)

  if F is a spanning tree then

  return MBST(GB,w)

  else

  return MBST((GA) η , w)

Camerini's Algorithm 1

- The above algorithm runs in $O(m)$ time

  UH, FOREST, GB all take $O(m/2^i)$ steps at i-th iteration

  $O(m + m/2 + m/4 + \ldots + 1) = O(m)$

Camerini's Algorithm 2

- What if the graph is directed?
- Definition

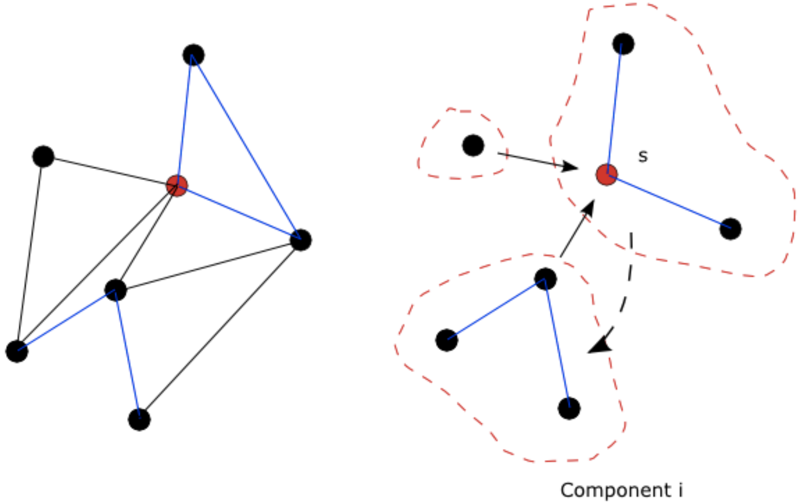
  G = (V, E) a directed graph

  s = a distinguished root vertex of G
- MBST

  a spanning tree rooted at s (containing paths from s to all other vertices) whose bottleneck edge weight is minimum.

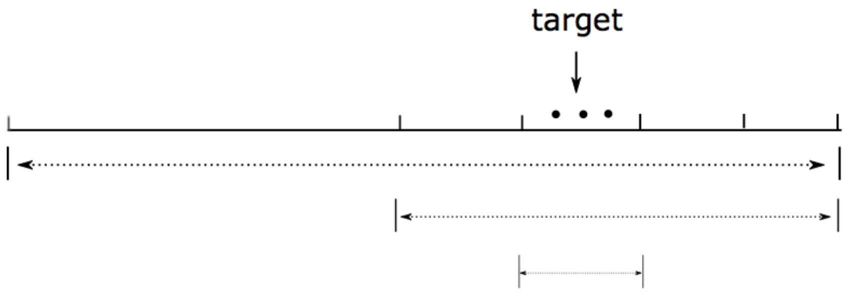
Camerini's Algorithm 2

- Theorem 1 (b) does not hold true



Camerini's Algorithm 2

- A verification for the path from root vertex s to all vertices is needed
- Theorem 1 (a) still works !
- Find bottleneck weight



Camerini's Algorithm 2

- E := the set of edges of G

  if | E - T | > 1 then

  A := UH(E - T, w)

  B := (E – T) - A

  F := BUSH (G{TUB})

  if F is a spanning tree then

  S := F; MBST(G{TUB} ,w, T)

  else

  MBST(G, w, TUB)

Camerini's Algorithm 2

- The above algorithm runs in O(mlogn) time

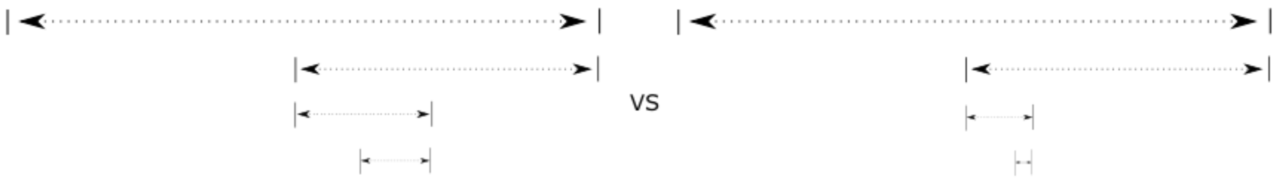
  UH, BUSH both take O(m) steps at i-th iteration

  Half partition, O(logm) < O(2logn) ~ O(logn) iterations

Gabow and Tarjan’s Algorithm

- Half partition
- Can it be faster?
- A special partition

  Less iterations…



Gabow and Tarjan's Algorithm

- 1. $i := i+1$ $S0 := \{(v,w) \in E \mid c(v, w) \leq \lambda 1\}$ $E1 := \{(v, w) \in E \mid \lambda 1 < c(v, w) \leq \lambda 2\}$
- 2. Partition E1 into k(i) subsets S1, S2, ..., Sk(i), each of size about $|E1| / k(i)$, such that if $(v, w) \in Si$ and $(x, y) \in Si+1$ then $c(v, w) \leq c(x, y)$.
- 3. Find the minimum j such that $Gj = (V, S0 \cup S1 \cup S2 \cup \cdots \cup Sj)$ is such that all vertices are reachable from s
- 4. if $j = 0$ then $\lambda * = \lambda 1$ stop, else replace $\lambda 1$ and $\lambda 2$, respectively, by the minimum cost and the maximum cost of an edge in Sj, go to step 1

Gabow and Tarjan's Algorithm

- The above algorithm runs in $O(m\log{*}n)$ time

  Steps 1, 3, 4 in $O(m)$ time

  Define $k(1) = 2$, $k(i) = 2$^$k(i-1)$

  $|E1| = O(m/k(i-1))$

  Steps 2 takes $O(m/k(i-1) * \log k(i)) = O(m)$ time

  However, only needs $O(\log{*}n)$ iterations

Thank you

Welcome your questions and comments