Minumum Cut

Stoer and Wagner's deterministic Algorithm

Advanced Algorithms

Davide Bresolin

2021/2022

Minimum cut for weighted graphs

Problem

Given a weighted undirected graph $G = (V, E, w)$ with positive weights, and a cut (S, T) of G, the weight of the cut (S, T) is defined as

$$w(S, T) = \sum_{e \in E(S,T)} w(e).$$

The global min-cut problem is the problem of finding a cut of G with the smallest weight.

Variant: s, t minimum cut

Given two vertices $s, t \in V$, an s, t minimum cut is a cut (S, T) of G such that:

- $s \in S$ and $t \in T$ (the cut separates s from t)
- the weight $w(S, T)$ is the smallest possible among all s, t cuts

Stoer and Wagner's algorithm: idea

Remark

Let (S, T) be a global min-cut of G. For every pair of nodes $s, t \in V$, two cases are possible:

- either (S, T) separates s from t, or
- s and t are on the same side of the cut

Stoer and Wagner's algorithm

1. find an s, t min-cut (S, T) of G, for some two vertices $s, t \in V$
2. two cases:
   - (S, T) is also a global min-cut
   - in any global-min cut of G, s and t are on the same side of the cut
3. in the second case a global min-cut of $G/\{s, t\}$ is also a global min-cut of G

Stoer and Wagner's algorithm: general structure

function GlobalMinCut($G = (V, E, w)$)
    if $V = \{a, b\}$ **then**
        return $(\{a\}, \{b\})$
    else
        $(C_1, s, t) \leftarrow \text{stMinCut}(G)$ *// C_1 is an s, t min-cut*
        $C_2 \leftarrow \text{GlobalMinCut}(G/\{s, t\})$
        if $w(C_1) \leq w(C_2)$ **then**
            return C_1
        else
            return C_2

How to find an s, t min-cut

- Finding an s, t min-cut is harder than finding a global min-cut, when s and t are specified
- Here we do not care who s and t are: they are not passed to stMinCut: they are returned by the function
- Finding a cut (S, T) and two vertices s, t such (S, T) is an s, t min-cut is a much easier problem!

stMinCut

function stMinCut($G = (V, E, w)$)
    $A \leftarrow \{a\}$
    while $A \neq V$ **do**
        Let $v \in V$ be such that $w(A, \{v\})$ is maximized
        $A \leftarrow A \cup \{v\}$
    Let s and t be the last two vertices added to A
    return $(V - \{t\}, \{t\}), s, t$

Lemma (stMinCut correctness)

If s and t are the last two vertices added to A by stMinCut, *then $(V - \{t\}, \{t\})$ is an s, t min-cut of G.*

How to implement stMinCut

function stMinCut($G = (V, E, w)$)
  $Q \leftarrow \emptyset$ // *Q priority queue*
  for all $u \in V$ **do**
    $key[u] \leftarrow 0$
    Insert($Q, u, key[u]$)
  $s, t \leftarrow NULL$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ ExtractMax(Q)
    $s \leftarrow t$; $t \leftarrow u$
    for all $v \in Adj[u]$ **do**
      if $v \in Q$ **then**
        $key[v] \leftarrow key[v] + w(u, v)$
        IncreaseKey($Q, v, key[v]$)
  return $(V - \{t\}, \{t\}), s, t$

Complexity

Given a graph G with n vertices and m edges:

- the execution time of stMinCut depends on the implementation of Q:
  - implemented with a MaxHeap: $O(m \log n)$
  - implemented with a Fibonacci Heap: $O(m + n \log n)$
- the execution time of GlobalMinCut is:
  - priority queue implemented with a MaxHeap: $O(mn \log n)$
  - prioirity queue implemented with a Fibonacci Heap: $O(mn + n^2 \log n)$