Streaming Algorithms

Meng-Tsung Tsai

05/01/2018

References

- "On Graph Problems in a Semi-Streaming Model," Feigenbaum et al. (2005)

Matching

Problem Definition

Input: a sequence of edges $e_1, e_2, \ldots, e_m$ of an n-node m-edge undirected graph G.

Output: a matching M of G.

Goal: make M as large as possible using O(n poly log n) space and few passes.

1/2-approximation, on a RAM

```
Matching(G){

  M ← ∅;
  foreach(e){
     if(M ∪ {e} is a matching){
        M ← M ∪ {e};
     }
  }
  return M;
}
```

$|M| \geq (1/2)$OPT. (Why?)

2/3-approximation, on a RAM

```
Matching(G){

  M ← ∅;
  foreach(e){
     if(M ∪ {e} is a matching){
        M ← M ∪ {e};
     }
  }
  while(there exists a length-3 augmenting path P w.r.t. M){
     M ← M ⊕ P;
  }

  return M;
}
```

$|M| \geq (2/3)$OPT. (Why?)

Exercise 1

Devise an analogous algorithm that yields a $(1-\varepsilon)$-approximation for matching for any constant $\varepsilon > 0$.

$(2/3-\varepsilon)$-approximation for bipartite graphs in the semi-streaming model

Augmentable length-3 paths

Let X be a maximum-size set of simultaneously augmentable length-3 paths with resprect to M.

$|M|+|X| \geq (2/3)$OPT. (Why?)

Augmentable length-3 paths

Let X be a maximum-size set of simultaneously augmentable length-3 paths with resprect to M.

$|M|+|X| \geq (2/3)$OPT. (Why?)

Let X' be a maximal-size set of simultaneously augmentable length-3 paths with represent to M.

$|X'| \geq (1/3)|X|$. (Why?)

Semi-streaming Algorithm (First Attempt)

```
Matching(G){
  M ← any maximal matching of G;

  for(r = 1; r ≤ 1/(2ε); ++r){
     X_r ← any (1/3)-apx of X;
     M ← M ⊕ X_r;
  }

  return M;
}
```

Matching(G) needs $O(1/(2\varepsilon))$ passes if X can be (1/3)-approximated $O(1)$ passes, after which $|M| \geq (2/3-\varepsilon)$OPT. (Why?)

Semi-streaming Algorithm (Second Attempt)

```
Matching(G){
  M ← any maximal matching of G;

  for(r = 1; r ≤ ⌈log 6ε / log 8/9⌉; ++r){
     X_r ← any ((1-2δ)/3)-apx of X; // δ = ε/(2-3ε)
     M ← M ⊕ X_r;
  }

  return M;
}
```

Matching(G) needs $O(\log(1/\varepsilon)/\varepsilon)$ passes if X can be $((1-2\delta)/3)$-approximated in $O(1/\delta)$ passes, after which $|M| \geq (2/3-\varepsilon)$OPT. (See the calculation in Ref. 1.)

$((1-2\boldsymbol{\delta})/3)$-apx of X

Approximating-X(M, $\boldsymbol{\delta}$){ // order all edges in M from one partite to the other
$V_C \leftarrow \varnothing$; // used nodes
$P_3 \leftarrow \varnothing$; // length-3 augmenting paths
while(1){
$P_2 \leftarrow \varnothing$; // length-2 augmenting paths
foreach(edge $(x, u) \in E/M$){
if($\exists$ ordered $(u, v) \in M$ and $x, u, v \notin V_C$)
$V_C \leftarrow V_C \cup \{x, u, v\}$, $P_2 \leftarrow P_2 \cup \{(x, u, v)\}$;
}
if ($|P_2| \le \boldsymbol{\delta}|M|$) return P_3; // $|X| \le 2\boldsymbol{\delta}|M|$
foreach(edge $(v, y) \in E/M$){
if($\exists$ ordered $(x, u, v) \in P_2$ and $x, u, v, y \notin V_C$)
$V_C \leftarrow V_C \cup \{x, u, v, y\}$, $P_3 \leftarrow P_3 \cup \{(x, u, v, y)\}$;
}
}
return P_3;
}

O($2/\boldsymbol{\delta}$) passes because each round except the last marks at least $\boldsymbol{\delta}|M|$ nodes, and each round uses 2 passes.

$((1-2\boldsymbol{\delta})/3)$-apx of X

Approximating-X(M, $\boldsymbol{\delta}$){ // order all edges in M from one partite to the other
$V_C \leftarrow \varnothing$; // used nodes
$P_3 \leftarrow \varnothing$; // length-3 augmenting paths
while(1){
$P_2 \leftarrow \varnothing$; // length-2 augmenting paths
foreach(edge $(x, u) \in E/M$){
if($\exists$ ordered $(u, v) \in M$ and $x, u, v \notin V_C$)
$V_C \leftarrow V_C \cup \{x, u, v\}$, $P_2 \leftarrow P_2 \cup \{(x, u, v)\}$;
}
if ($|P_2| \le \boldsymbol{\delta}|M|$) return P_3; // $|X| \le 2\boldsymbol{\delta}|M|$
foreach(edge $(v, y) \in E/M$){
if($\exists$ ordered $(x, u, v) \in P_2$ and $x, u, v, y \notin V_C$)
$V_C \leftarrow V_C \cup \{x, u, v, y\}$, $P_3 \leftarrow P_3 \cup \{(x, u, v, y)\}$;
}
}
return P_3;
}

Every length-3 path in P_3 kills 3 paths in X.

$((1-2\boldsymbol{\delta})/3)$-apx of X

Approximating-X(M, $\boldsymbol{\delta}$){ // order all edges in M from one partite to the other
$V_C \leftarrow \varnothing$; // used nodes
$P_3 \leftarrow \varnothing$; // length-3 augmenting paths
while(1){
$P_2 \leftarrow \varnothing$; // length-2 augmenting paths
foreach(edge $(x, u) \in E/M$){
if($\exists$ ordered $(u, v) \in M$ and $x, u, v \notin V_C$)
$V_C \leftarrow V_C \cup \{x, u, v\}$, $P_2 \leftarrow P_2 \cup \{(x, u, v)\}$;
}
if ($|P_2| \le \boldsymbol{\delta}|M|$) return P_3; // $|X| \le 2\boldsymbol{\delta}|M|$
foreach(edge $(v, y) \in E/M$){
if($\exists$ ordered $(x, u, v) \in P_2$ and $x, u, v, y \notin V_C$)
$V_C \leftarrow V_C \cup \{x, u, v, y\}$, $P_3 \leftarrow P_3 \cup \{(x, u, v, y)\}$;
}
}
return P_3;
}

Every length-2 path in P_2 kills 2 paths in X.