Planar Separator Theorem

Aleksandar Timanov

18.05.2018

Motivation

1. Dividing the problem recursively into smaller ones of the same type can speed up their solving substantially.
2. Some already known examples may be:
   - Merge sort
   - Quicksort
3. Enables for even quicker solution using parallel programmig to solve the subproblems simultaneously.
4. Easier memory access, because a small enough subproblem can be solved only using the cache for example.

Motivation

1. Dividing the problem recursively into smaller ones of the same type can speed up their solving substantially.
2. Some already known examples may be:
   - Merge sort
   - Quicksort
3. Enables for even quicker solution using parallel programmig to solve the subproblems simultaneously.
4. Easier memory access, because a small enough subproblem can be solved only using the cache for example.

Motivation

1. Dividing the problem recursively into smaller ones of the same type can speed up their solving substantially.
2. Some already known examples may be:
   - Merge sort
   - Quicksort
3. Enables for even quicker solution using parallel programmig to solve the subproblems simultaneously.
4. Easier memory access, because a small enough subproblem can be solved only using the cache for example.

Motivation

1. Dividing the problem recursively into smaller ones of the same type can speed up their solving substantially.
2. Some already known examples may be:
   - Merge sort
   - Quicksort
3. Enables for even quicker solution using parallel programmig to solve the subproblems simultaneously.
4. Easier memory access, because a small enough subproblem can be solved only using the cache for example.

Motivation

1. Dividing the problem recursively into smaller ones of the same type can speed up their solving substantially.
2. Some already known examples may be:
   - Merge sort
   - Quicksort
3. Enables for even quicker solution using parallel programmig to solve the subproblems simultaneously.
4. Easier memory access, because a small enough subproblem can be solved only using the cache for example.

Motivation

1. Dividing the problem recursively into smaller ones of the same type can speed up their solving substantially.
2. Some already known examples may be:
   - Merge sort
   - Quicksort
3. Enables for even quicker solution using parallel programmig to solve the subproblems simultaneously.
4. Easier memory access, because a small enough subproblem can be solved only using the cache for example.

Some useful theorems

Theorem 1

Any closed cureve in the plane devides it into an "inside" and "outside" region

Theorem 2 (Kurtakowski's theorem)

A graph is planar if and only if it contains neither K_5 or $K_{3,3}$ as a generalized subgraph

Theorem 3

Shrinking any edge to a sindle vertex preserves planarity.

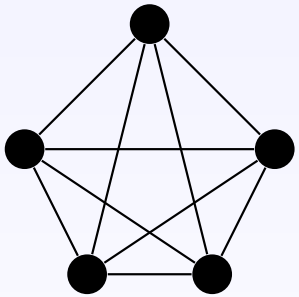
Some useful theorems

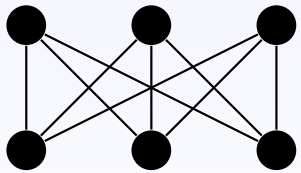
Theorem 1

Any closed cureve in the plane devides it into an "inside" and "outside" region

Theorem 2 (Kurtakowski's theorem)

A graph is planar if and only if it contains neither K_5 or $K_{3,3}$ as a generalized subgraph





Theorem 3

Shrinking any edge to a sindle vertex preserves planarity.

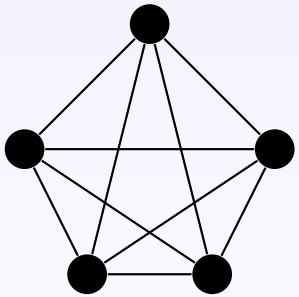
Some useful theorems

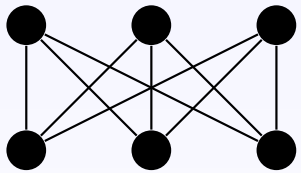
Theorem 1

Any closed cureve in the plane devides it into an "inside" and "outside" region

Theorem 2 (Kurtakowski's theorem)

A graph is planar if and only if it contains neither K_5 or $K_{3,3}$ as a generalized subgraph





Theorem 3

Shrinking any edge to a sindle vertex preserves planarity.

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Lemma 1

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more than one
- G contains a spanning tree with radius r

Result

then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $2r + 1$ vertices

Proof of Lemma 1

1. Embed G in the plane.
2. Triangulate.
3. Every nontree edge forms a simple cycle with some of the tree edges.
4. This cycle devides the plane into 2 parts. (Theorem 1)
5. At least one of those cycles separates the graph into parts, whose cost does not exceed $2/3$
6. The length of this cycles is at most $2r + 1$ where r is the radius of the spanning three.

Proof of Lemma 1

1. Embed G in the plane.
2. Triangulate.
3. Every nontree edge forms a simple cycle with some of the tree edges.
4. This cycle devides the plane into 2 parts. (Theorem 1)
5. At least one of those cycles separates the graph into parts, whose cost does not exceed $2/3$
6. The length of this cycles is at most $2r + 1$ where r is the radius of the spanning three.

Proof of Lemma 1

1. Embed G in the plane.
2. Triangulate.
3. Every nontree edge forms a simple cycle with some of the tree edges.
4. This cycle devides the plane into 2 parts. (Theorem 1)
5. At least one of those cycles separates the graph into parts, whose cost does not exceed $2/3$
6. The length of this cycles is at most $2r + 1$ where r is the radius of the spanning three.

Proof of Lemma 1

1. Embed G in the plane.
2. Triangulate.
3. Every nontree edge forms a simple cycle with some of the tree edges.
4. This cycle devides the plane into 2 parts. (Theorem 1)
5. At least one of those cycles separates the graph into parts, whose cost does not exceed $2/3$
6. The length of this cycles is at most $2r + 1$ where r is the radius of the spanning three.

Proof of Lemma 1

1. Embed G in the plane.
2. Triangulate.
3. Every nontree edge forms a simple cycle with some of the tree edges.
4. This cycle devides the plane into 2 parts. (Theorem 1)
5. At least one of those cycles separates the graph into parts, whose cost does not exceed $2/3$
6. The length of this cycles is at most $2r + 1$ where r is the radius of the spanning three.

Proof of Lemma 1

1. Embed G in the plane.
2. Triangulate.
3. Every nontree edge forms a simple cycle with some of the tree edges.
4. This cycle devides the plane into 2 parts. (Theorem 1)
5. At least one of those cycles separates the graph into parts, whose cost does not exceed $2/3$
6. The length of this cycles is at most $2r + 1$ where r is the radius of the spanning three.

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r + 1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r + 1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r+1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r + 1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r+1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r+1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding 2/3
  - Vertices on levels $l_2 + 1$ through $r + 1$ have total cost not exceeding 2/3

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding $2/3$
  - Vertices on levels $l_2 + 1$ through $r + 1$ have total cost not exceeding $2/3$

Lemma 2 (Leveling)

Given

- G is a connected planar graph
- G's vertices have nonnegative costs summing to no more then one
- G's vertices are partitioned into levels according to their distance from some vertex v
  - $L(l)$ is the number of vertices on level l
  - r (radius) is the maximum distance from v
  - Let $r+1$ be an additional layer without any vertices
- Given l_1 and l_2 such that
  - Vertices on levels 0 through $l_1 - 1$ have total cost not exceeding $2/3$
  - Vertices on levels $l_2 + 1$ through $r+1$ have total cost not exceeding $2/3$

Lemma 2

Result

Then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $L(l_1) + L(l_2) + max\{0, 2(l_2 - l_1 - 1)\}$ vertices

Lemma 2

Result

Then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $L(l_1) + L(l_2) + max\{0, 2(l_2 - l_1 - 1)\}$ vertices

Lemma 2

Result

Then we can partition the vertices so that:

- no edges connect A and B
- A and B both cost at most $2/3$
- C contains at most $L(l_1) + L(l_2) + max\{0, 2(l_2 - l_1 - 1)\}$ vertices

Proof of Lemma 2

1. If $l_1 \geq l_2$ then the lemma is true
   - A would contain the vertices on levels 0 through $l_1 - 1$ (cost is less than $2/3$)
   - B would contain those on levels $l_1 + 1$ through r (cost is less than $2/3$)
   - C would contain the vertices on l_1 (contains $L(l_1)$)

Proof of Lemma 2

1. If $l_1 \geq l_2$ then the lemma is true
   - A would contain the vertices on levels 0 through $l_1 - 1$ (cost is less than $2/3$)
   - B would contain those on levels $l_1 + 1$ through r (cost is less than $2/3$)
   - C would contain the vertices on l_1 (contains $L(l_1)$)

Proof of Lemma 2

1. If $l_1 \geq l_2$ then the lemma is true
   - A would contain the vertices on levels 0 through $l_1 - 1$ (cost is less than $2/3$)
   - B would contain those on levels $l_1 + 1$ through r (cost is less than $2/3$)
   - C would contain the vertices on l_1 (contains $L(l_1)$)

Proof of Lemma 2

2. If $l_1 < l_2$ the lemma still hold true
   - Delete all vertices on levels l_1 and l_2, partitioning the graph into 3 part. (Only the middle one can have a cost exceeding 2/3)
   - If it costs more than 2/3:
     - delete all the vertices on l_2 and above
     - shrink those on l_1 and below to a single one.
   - Then the new graph contains a spanning tree with radius $l_2 - 1 - l_1$ and we can apply lemma 2, which results in A^*, B^*, C^*
   - Then A would be assigned A^* or B^* depending on which has higher cost.
     C gets the vertices on l_1 and l_2 as well as C^*.
     And B is the rest.

Proof of Lemma 2

2. If $l_1 < l_2$ the lemma still hold true
   - Delete all vertices on levels l_1 and l_2, partitioning the graph into 3 part. (Only the middle one can have a cost exceeding 2/3)
   - If it costs more than 2/3:
     - delete all the vertices on l_2 and above
     - shrink those on l_1 and below to a single one.
   - Then the new graph contains a spanning tree with radius $l_2 - 1 - l_1$ and we can apply lemma 2, which results in A^*, B^*, C^*
   - Then A would be assigned A^* or B^* depending on which has higher cost.
     C gets the vertices on l_1 and l_2 as well as C^*.
     And B is the rest.

Proof of Lemma 2

2. If $l_1 < l_2$ the lemma still hold true
   - Delete all vertices on levels l_1 and l_2, partitioning the graph into 3 part. (Only the middle one can have a cost exceeding 2/3)
   - If it costs more than 2/3:
     - delete all the vertices on l_2 and above
     - shrink those on l_1 and below to a single one.
   - Then the new graph contains a spanning tree with radius $l_2 - 1 - l_1$ and we can apply lemma 2, which results in A^*, B^*, C^*
   - Then A would be assigned A^* or B^* depending on which has higher cost.
     C gets the vertices on l_1 and l_2 as well as C^*.
     And B is the rest.

Proof of Lemma 2

2. If $l_1 < l_2$ the lemma still hold true
   - Delete all vertices on levels l_1 and l_2, partitioning the graph into 3 part. (Only the middle one can have a cost exceeding 2/3)
   - If it costs more than 2/3:
     - delete all the vertices on l_2 and above
     - shrink those on l_1 and below to a single one.
   - Then the new graph contains a spanning tree with radius $l_2 - 1 - l_1$ and we can apply lemma 2, which results in A^*, B^*, C^*
   - Then A would be assigned A^* or B^* depending on which has higher cost.
     C gets the vertices on l_1 and l_2 as well as C^*.
     And B is the rest.

Proof of Lemma 2

2. If $l_1 < l_2$ the lemma still hold true
   - Delete all vertices on levels l_1 and l_2, partitioning the graph into 3 part. (Only the middle one can have a cost exceeding $2/3$)
   - If it costs more than $2/3$:
     - delete all the vertices on l_2 and above
     - shrink those on l_1 and below to a single one.
   - Then the new graph contains a spanning tree with radius $l_2 - 1 - l_1$ and we can apply lemma 2, which results in A^*, B^*, C^*
   - Then A would be assigned A^* or B^* depending on which has higher cost.
     C gets the vertices on l_1 and l_2 as well as C^*.
     And B is the rest.

Proof of Lemma 2

2. If $l_1 < l_2$ the lemma still hold true
   - Delete all vertices on levels l_1 and l_2, partitioning the graph into 3 part. (Only the middle one can have a cost exceeding $2/3$)
   - If it costs more than $2/3$:
     - delete all the vertices on l_2 and above
     - shrink those on l_1 and below to a single one.
   - Then the new graph contains a spanning tree with radius $l_2 - 1 - l_1$ and we can apply lemma 2, which results in A^*, B^*, C^*
   - Then A would be assigned A^* or B^* depending on which has higher cost.
     C gets the vertices on l_1 and l_2 as well as C^*.
     And B is the rest.

Theorem 4

Given

- G is a n-vertex planar graph
- G's vertices have nonnegative costs summing to no more than one

Result

Then G's vertices can be partiotioned into the sets A, B and C so that:

- A and B both cost at most $2/3$
- C contains no more than $2\sqrt{2}\sqrt{n}$ vertices

Proof of theorem 4

1. If G is connected, the theorem is obviously true, because we can layer the graph and apply lemma 2
2. If G is not connected(Let $G_1, G_2, G_3..., G_k$ with vertices $V_1, V_2, ..., V_k$ respectively). Then we have the following cases:
   - No component has cost exceeding 1/3. Then let i be the minimal index so that $A = V_1 \cup V_2 \cup ... \cup V_i$'s cost exceeds 1/3. Then $B = V_{i+1} \cup V_{i+2} \cup ... \cup V_k$ and $C = \emptyset$
   - If some G_i has cost between 1/3 and 2/3, then we simply make $A = V_i$, B would be left with all the other vertices that arent in A and $C = \emptyset$
   - If some G_i's cost exceeds 2/3 we do step 1, resulting in A^*, B^*, C^*. Then let A be the greater between A^* and B^*, $C = C^*$ and B be the rest.
     ($A = \frac{2}{3} * \frac{2}{3} = \frac{4}{9} < \frac{6}{9} = \frac{2}{3}$ in the worst case and $B = \frac{5}{9} < \frac{6}{9}$ in this case)

Proof of theorem 4

1. If G is connected, the theorem is obviously true, because we can layer the graph and apply lemma 2
2. If G is not connected(Let $G_1, G_2, G_3..., G_k$ with vertices $V_1, V_2, ..., V_k$ respectively). Then we have the following cases:
   - No component has cost exceeding 1/3. Then let i be the minimal index so that $A = V_1 \cup V_2 \cup ... \cup V_i$'s cost exceeds 1/3. Then $B = V_{i+1} \cup V_{i+2} \cup ... \cup V_k$ and $C = \varnothing$
   - If some G_i has cost between 1/3 and 2/3, then we simply make $A = V_i$, B would be left with all the other vertices that arent in A and $C = \varnothing$
   - If some G_i's cost exceeds 2/3 we do step 1, resulting in A^*, B^*, C^*. Then let A be the greater between A^* and B^*, $C = C^*$ and B be the rest.
     ($A = \frac{2}{3} * \frac{2}{3} = \frac{4}{9} < \frac{6}{9} = \frac{2}{3}$ in the worst case and $B = \frac{5}{9} < \frac{6}{9}$ in this case)

Proof of theorem 4

1. If G is connected, the theorem is obviously true, because we can layer the graph and apply lemma 2
2. If G is not connected(Let $G_1, G_2, G_3..., G_k$ with vertices $V_1, V_2, ..., V_k$ respectively). Then we have the following cases:

   - No component has cost exceeding $1/3$. Then let i be the minimal index so that $A = V_1 \cup V_2 \cup ... \cup V_i$'s cost exceeds $1/3$. Then $B = V_{i+1} \cup V_{i+2} \cup ... \cup V_k$ and $C = \varnothing$
   - If some G_i has cost between $1/3$ and $2/3$, then we simply make $A = V_i$, B would be left with all the other vertices that arent in A and $C = \varnothing$
   - If some G_i's cost exceeds $2/3$ we do step 1, resulting in A^*, B^*, C^*. Then let A be the greater between A^* and B^*, $C = C^*$ and B be the rest.
     ($A = \frac{2}{3} * \frac{2}{3} = \frac{4}{9} < \frac{6}{9} = \frac{2}{3}$ in the worst case and $B = \frac{5}{9} < \frac{6}{9}$ in this case)

Proof of theorem 4

1. If G is connected, the theorem is obviously true, because we can layer the graph and apply lemma 2
2. If G is not connected(Let $G_1, G_2, G_3..., G_k$ with vertices $V_1, V_2, ..., V_k$ respectively). Then we have the following cases:

   - No component has cost exceeding $1/3$. Then let i be the minimal index so that $A = V_1 \cup V_2 \cup ... \cup V_i$'s cost exceeds $1/3$. Then $B = V_{i+1} \cup V_{i+2} \cup ... \cup V_k$ and $C = \varnothing$
   - If some G_i has cost between $1/3$ and $2/3$, then we simply make $A = V_i$, B would be left with all the other vertices that arent in A and $C = \varnothing$
   - If some G_i's cost exceeds $2/3$ we do step 1, resulting in A^*, B^*, C^*. Then let A be the greater between A^* and B^*, $C = C^*$ and B be the rest.
     ($A = \frac{2}{3} * \frac{2}{3} = \frac{4}{9} < \frac{6}{9} = \frac{2}{3}$ in the worst case and $B = \frac{5}{9} < \frac{6}{9}$ in this case)

Proof of theorem 4

1. If G is connected, the theorem is obviously true, because we can layer the graph and apply lemma 2
2. If G is not connected(Let $G_1, G_2, G_3..., G_k$ with vertices $V_1, V_2, ..., V_k$ respectively). Then we have the following cases:

   - No component has cost exceeding $1/3$. Then let i be the minimal index so that $A = V_1 \cup V_2 \cup ... \cup V_i$'s cost exceeds $1/3$. Then $B = V_{i+1} \cup V_{i+2} \cup ... \cup V_k$ and $C = \varnothing$
   - If some G_i has cost between $1/3$ and $2/3$, then we simply make $A = V_i$, B would be left with all the other vertices that arent in A and $C = \varnothing$
   - If some G_i's cost exceeds $2/3$ we do step 1, resulting in A^*, B^*, C^*. Then let A be the greater between A^* and B^*, $C = C^*$ and B be the rest.
     ($A = \frac{2}{3} * \frac{2}{3} = \frac{4}{9} < \frac{6}{9} = \frac{2}{3}$ in the worst case and $B = \frac{5}{9} < \frac{6}{9}$ in this case)

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed 2/3 we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under 1/2
   - the total cost on levels 0 to l_1 is over 1/2
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed 2/3 we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under 1/2
   - the total cost on levels 0 to l_1 is over 1/2
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed 2/3 we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under 1/2
   - the total cost on levels 0 to l_1 is over 1/2
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed 2/3 we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under 1/2
   - the total cost on levels 0 to l_1 is over 1/2
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed 2/3 we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under 1/2
   - the total cost on levels 0 to l_1 is over 1/2
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed 2/3 we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under 1/2
   - the total cost on levels 0 to l_1 is over 1/2
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed $2/3$ we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under $1/2$
   - the total cost on levels 0 to l_1 is over $1/2$
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed $2/3$ we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under $1/2$
   - the total cost on levels 0 to l_1 is over $1/2$
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

1. Find a **planar embedding** of G
2. Find the **connected componenents** and determine the cost of each one.
   - If none of the components costs exceed $2/3$ we can apply Theorem 4 and be done.
   - Else we have to go to step 3.
3. Do a breadth-first **spanning tree** of the most costly component. Compute the **level of each vertex** and the number of vertices on each level.
4. Find **level l_1** such that
   - the total cost on levels 0 to $l_1 - 1$ is under $1/2$
   - the total cost on levels 0 to l_1 is over $1/2$
   - let k be the number of vertices on levels 0 to l_1

The Algorithm

5. Find the **highest level** $l_0 \leq l_1$ such that $L(l_0) + 2(l_1 - l_0) \leq 2\sqrt{k}$. Find the **smallest level** $l_2 \geq l_1 + 1$ such that $L(l_2) + 2(l_2 - l_1 - 1) \leq 2\sqrt{n-k}$
6. **Delete** all vertices on levels l_2 and above. **Shrink** all vertices on levels 0 through l_0.
7. Do a breadth-first **spanning tree** rooted at x. Record all the parent of each vertex and the total cost of all descendents + the cost of the vertex itself. **Triangulate**.
8. Let (v_1, w_1) be a nontree edge and find a **cycle** by following parents. Find the cost "inside" and "outside" the cycle by scanning the .
9. Optimize cycle if the cost "inside" is over 2/3.
10. Apply **Theorem 4**.

The Algorithm

5. Find the **highest level** $l_0 \leq l_1$ such that $L(l_0) + 2(l_1 - l_0) \leq 2\sqrt{k}$. Find the **smallest level** $l_2 \geq l_1 + 1$ such that $L(l_2) + 2(l_2 - l_1 - 1) \leq 2\sqrt{n-k}$
6. **Delete** all vertices on levels l_2 and above. **Shrink** all vertices on levels 0 through l_0.
7. Do a breadth-first **spanning tree** rooted at x. Record all the parent of each vertex and the total cost of all descendents + the cost of the vertex itself. **Triangulate**.
8. Let (v_1, w_1) be a nontree edge and find a **cycle** by following parents. Find the cost "inside" and "outside" the cycle by scanning the .
9. Optimize cycle if the cost "inside" is over 2/3.
10. Apply **Theorem 4**.

The Algorithm

5. Find the **highest level** $l_0 \leq l_1$ such that $L(l_0) + 2(l_1 - l_0) \leq 2\sqrt{k}$. Find the **smallest level** $l_2 \geq l_1 + 1$ such that $L(l_2) + 2(l_2 - l_1 - 1) \leq 2\sqrt{n-k}$
6. **Delete** all vertices on levels l_2 and above. **Shrink** all vertices on levels 0 through l_0.
7. Do a breadth-first **spanning tree** rooted at x. Record all the parent of each vertex and the total cost of all descendents + the cost of the vertex itself. **Triangulate**.
8. Let (v_1, w_1) be a nontree edge and find a **cycle** by following parents. Find the cost "inside" and "outside" the cycle by scanning the .
9. Optimize cycle if the cost "inside" is over 2/3.
10. Apply **Theorem 4**.

The Algorithm

5. Find the **highest level** $l_0 \leq l_1$ such that $L(l_0) + 2(l_1 - l_0) \leq 2\sqrt{k}$. Find the **smallest level** $l_2 \geq l_1 + 1$ such that $L(l_2) + 2(l_2 - l_1 - 1) \leq 2\sqrt{n-k}$
6. **Delete** all vertices on levels l_2 and above. **Shrink** all vertices on levels 0 through l_0.
7. Do a breadth-first **spanning tree** rooted at x. Record all the parent of each vertex and the total cost of all descendents + the cost of the vertex itself. **Triangulate**.
8. Let (v_1, w_1) be a nontree edge and find a **cycle** by following parents. Find the cost "inside" and "outside" the cycle by scanning the .
9. Optimize cycle if the cost "inside" is over 2/3.
10. Apply **Theorem 4**.

The Algorithm

5. Find the **highest level** $l_0 \leq l_1$ such that $L(l_0) + 2(l_1 - l_0) \leq 2\sqrt{k}$. Find the **smallest level** $l_2 \geq l_1 + 1$ such that $L(l_2) + 2(l_2 - l_1 - 1) \leq 2\sqrt{n-k}$
6. **Delete** all vertices on levels l_2 and above. **Shrink** all vertices on levels 0 through l_0.
7. Do a breadth-first **spanning tree** rooted at x. Record all the parent of each vertex and the total cost of all descendents + the cost of the vertex itself. **Triangulate**.
8. Let (v_1, w_1) be a nontree edge and find a **cycle** by following parents. Find the cost "inside" and "outside" the cycle by scanning the .
9. Optimize cycle if the cost "inside" is over 2/3.
10. Apply **Theorem 4**.

The Algorithm

5. Find the **highest level** $l_0 \leq l_1$ such that $L(l_0) + 2(l_1 - l_0) \leq 2\sqrt{k}$. Find the **smallest level** $l_2 \geq l_1 + 1$ such that $L(l_2) + 2(l_2 - l_1 - 1) \leq 2\sqrt{n-k}$
6. **Delete** all vertices on levels l_2 and above. **Shrink** all vertices on levels 0 through l_0.
7. Do a breadth-first **spanning tree** rooted at x. Record all the parent of each vertex and the total cost of all descendents + the cost of the vertex itself. **Triangulate**.
8. Let (v_1, w_1) be a nontree edge and find a **cycle** by following parents. Find the cost "inside" and "outside" the cycle by scanning the .
9. Optimize cycle if the cost "inside" is over 2/3.
10. Apply **Theorem 4**.

Maximum independent set (NP-complete)

Theorem 5

Given is:

- Planar n-vertex graph G with nonnegative vertex costs
- $0 \leq \varepsilon \leq 1$

Then we can find:

- A set C of $\mathcal{O}(\sqrt{n/\varepsilon})$ whose removal leaves G with no connected component of cost exceeding ε

C can be found in $\mathcal{O}(n \log n)$

Maximum independent set (NP-complete)

Theorem 5

Given is:

- Planar n-vertex graph G with nonnegative vertex costs
- $0 \leq \varepsilon \leq 1$

Then we can find:

- A set C of $\mathcal{O}(\sqrt{n/\varepsilon})$ whose removal leaves G with no connected component of cost exceeding ε

C can be found in $\mathcal{O}(n \log n)$

Maximum independent set (NP-complete)

Theorem 5

Given is:

- Planar n-vertex graph G with nonnegative vertex costs
- $0 \leq \varepsilon \leq 1$

Then we can find:

- A set C of $\mathcal{O}(\sqrt{n/\varepsilon})$ whose removal leaves G with no connected component of cost exceeding ε

C can be found in $\mathcal{O}(n \log n)$

Maximum independent set (NP-complete)

Theorem 5

Given is:

- Planar n-vertex graph G with nonnegative vertex costs
- $0 \leq \varepsilon \leq 1$

Then we can find:

- A set C of $\mathcal{O}(\sqrt{n/\varepsilon})$ whose removal leaves G with no connected component of cost exceeding ε

C can be found in $\mathcal{O}(n \log n)$

Maximum independent set (NP-complete)

Theorem 5

Given is:

- Planar n-vertex graph G with nonnegative vertex costs
- $0 \leq \varepsilon \leq 1$

Then we can find:

- A set C of $\mathcal{O}(\sqrt{n/\varepsilon})$ whose removal leaves G with no connected component of cost exceeding ε

C can be found in $\mathcal{O}(n \log n)$

Maximum independent set (NP-complete)

Theorem 5

Given is:

- Planar n-vertex graph G with nonnegative vertex costs
- $0 \leq \varepsilon \leq 1$

Then we can find:

- A set C of $\mathcal{O}(\sqrt{n/\varepsilon})$ whose removal leaves G with no connected component of cost exceeding ε

C can be found in $\mathcal{O}(n \log n)$

Proof of Theorem 5

1. Find a separation of G by applying Theorem 4 until there are no more connected components with cost more than ε
2. The runtime is $\mathcal{O}(n \log n)$ as there we have $\mathcal{O}(\log n)$ separations each done in a linear time
3. C is $\mathcal{O}(\sqrt{n/\varepsilon})$ because we have a separator of size $\mathcal{O}(\sqrt{n})$ which has to be calculated for every separation (number of separations $= \mathcal{O}(n/\varepsilon)$)

Results

So using Theorem 5 for $\varepsilon = \log n/n$ and each vertex having cost $1/n$ the maximum independent set can be computed in $\mathcal{O}(n^2)$-time with an error rate of $\mathcal{O}(1/\sqrt{logn})$

Proof of Theorem 5

1. Find a separation of G by applying Theorem 4 until there are no more connected components with cost more than ε
2. The runtime is $\mathcal{O}(n \log n)$ as there we have $\mathcal{O}(\log n)$ separations each done in a linear time
3. C is $\mathcal{O}(\sqrt{n/\varepsilon})$ because we have a separator of size $\mathcal{O}(\sqrt{n})$ which has to be calculated for every separation (number of separations $= \mathcal{O}(n/\varepsilon)$)

Results

So using Theorem 5 for $\varepsilon = \log n/n$ and each vertex having cost $1/n$ the maximum independent set can be computed in $\mathcal{O}(n^2)$-time with an error rate of $\mathcal{O}(1/\sqrt{logn})$

Proof of Theorem 5

1. Find a separation of G by applying Theorem 4 until there are no more connected components with cost more than ε
2. The runtime is $\mathcal{O}(n \log n)$ as there we have $\mathcal{O}(\log n)$ separations each done in a linear time
3. C is $\mathcal{O}(\sqrt{n/\varepsilon})$ because we have a separator of size $\mathcal{O}(\sqrt{n})$ which has to be calculated for every separation (number of separations $= \mathcal{O}(n/\varepsilon)$)

Results

So using Theorem 5 for $\varepsilon = \log n/n$ and each vertex having cost $1/n$ the maximum independent set can be computed in $\mathcal{O}(n^2)$-time with an error rate of $\mathcal{O}(1/\sqrt{logn})$

Proof of Theorem 5

1. Find a separation of G by applying Theorem 4 until there are no more connected components with cost more than ε
2. The runtime is $\mathcal{O}(n \log n)$ as there we have $\mathcal{O}(\log n)$ separations each done in a linear time
3. C is $\mathcal{O}(\sqrt{n/\varepsilon})$ because we have a separator of size $\mathcal{O}(\sqrt{n})$ which has to be calculated for every separation (number of separations $= \mathcal{O}(n/\varepsilon)$)

Results

So using Theorem 5 for $\varepsilon = \log n/n$ and each vertex having cost $1/n$ the maximum independent set can be computed in $\mathcal{O}(n^2)$-time with an error rate of $\mathcal{O}(1/\sqrt{logn})$

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

Embedding data structures

Theorem 6

Any planar graph with maximum degree k can be embedded in a binary tree so that the average proximity is $\mathcal{O}(k)$

Proof:

1. If G has one vertex then T would be a tree of one vertex.
2. Otherwise we apply Theorem 4 with each vertex costing $1/n$, resulting in A, B, C
3. We choose one of the vertices in C and recursively embed the subgraph induced by $(A \cup C) \backslash \{v\}$ in a binary tree T_1
4. Recursively embed the subgraph induced by B in a binary tree T_2
5. Let T consist of a root (v's image) with two children, the root of T_1 and the root of T_2
6. By induction we can conclude that G can be embedded in $\mathcal{O}(k)$ time.

References I


Richard J. Lipton and Robert Endre Tarjan. A Separator Theorem for Planar Graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.


Richard J. Lipton and Robert Endre Tarjan. Applications of a Planar Separator Theorem. *SIAM Journal on Computing*, 9(3):615–627, 1980.