More Triangles! A Note on the Cactus-Tree Heuristic

Timo Poranen

University of Tampere
Department of Computer Sciences
33014 University of Tampere
email: tp@cs.uta.fi
Presentation structure


  
  – Suppose that a researcher has found a new algorithm for a problem. How to give a (scientific) description about the algorithm?

  
  – Manuscript describes a new approximation algorithm for the maximum planar subgraph problem.
Definitions

- A undirected, simple graph $G$ denoted $G = (V, E)$ consists of a finite vertex set $V$ and set of undirected edges $E \subseteq \{(v, u) \mid v \in V, u \in V, v \neq u\}$.

- Throughout this presentation, we consider only undirected simply graphs (i.e. there are no parallel edges or self-loops).

- Given a graph $G = (V, E)$, a graph $G' = (V', E')$ is called a subgraph of $G$ if $V' \subseteq V$ and $E' \subseteq \{(v, u) \mid v \in V', u \in V' \text{ and } (v, u) \in E\}$.

- Let $G = (V, E)$ be a graph and let $E' \subseteq E$. Then graph $G' = (V', E')$ is called an edge induced subgraph of $G$ if $V' = \{v \mid (v, u) \in E'\}$.

- If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are subgraphs of $G = (V, E)$, then subgraph $G' = (V_1 \cup V_2, E_1 \cup E_2)$ of $G$ is called the union of $G_1$ and $G_2$ (denoted $G_1 \cup G_2$).
Planar graphs

• A graph is usually visualized by representing each vertex through a point in the plane, and by representing each edge through a curve in the plane, connecting the points corresponding to the vertices of the edge.

• A drawing of a graph is planar if no two distinct edges intersect.

• A graph $G$ is planar if it admits a planar drawing. Such a drawing of a planar graph is called a planar embedding of $G$.

• A planar graph with $n$ vertices contains at most $3n - 6$ edges ($n \geq 3$).

• A graph $G$ is planar if and only if it does not contain a subgraph homeomorphic to $K_5$ or $K_{3,3}$ (Kuratowski 1930).
Planarity testing

• Testing whether a graph is planar or not can be done in linear time using classical planarity testing algorithm of Hopcroft-Tarjan.

• Hopcroft-Tarjan’s algorithm uses depth-first search and makes “some” calculations during it.

• There are more planarity testing algorithms:
  – Booth and Lueker’s algorithm which uses a data structure called \( PQ \)-trees.
  – A new algorithm of Shih and Hsu based on PC-trees.
Maximal planar subgraphs

• Let $G'$ be a subgraph of a graph $G$. If $G'$ is planar, then $G'$ is a planar subgraph of $G$.

• If a graph $G' = (V', E')$ is a planar subgraph of $G = (V, E)$ such that every graph $G'' \in (V, E' \cup \{e\} \mid e \in E \setminus E')$ is nonplanar, then $G'$ is called a maximal planar subgraph of $G$.

• The problem is solvable in polynomial time by using any (polynomial time) planarity testing algorithm.
Maximum planar subgraphs

- If a graph $G' = (V, E')$ is a planar subgraph of $G = (V, E)$ such that there is no planar subgraph $G'' = (V, E'')$ of $G$ with $|E''| > |E'|$, then $G$ is called a maximum planar subgraph of $G$.

- A maximal planar subgraph is maximal with respect to inclusion of its edge set, whereas a maximum planar subgraph is maximal with respect to the cardinality of its edge set.

- The problem of finding a maximum planar subgraph of given graph is $NP$-hard (Liu and Geldmacher 1977, Yannakakis 1978).

- Numerous approximation algorithms exists in the literature.

- Problem has applications in VLSI-circuit design, facility layout and graph drawing.
Examples from planarity and planar subgraphs

Figure 1: $G$ is a nonplanar graph. Note that $G$ contains $K_{3,3}$ as a minor. $G_1$ is a planar subgraph of $G$, but it is not a maximal planar subgraph: Edge $(1, 5)$ can be added to $G_1$ without destroying planarity. The result is $G_2$. Another maximal planar subgraph of $G$ is $G_3$. Graph $G_3$ is also a maximum planar subgraph.
Outerplanar graphs

• A planar graph is outerplanar if it admits a plane drawing where all its vertices lie on the same face and no two distinct edges intersect.

• Maximum and maximal outerplanar subgraphs are defined in the similar way as the maximum and maximal planar subgraphs.

• An outerplanar graph with $n$ vertices contains at most $2n - 3$ edges.

• A graph $G$ is outerplanar if and only if it does not contain a subgraph homeomorphic to $K_4$ or $K_{3,2}$ (Chartland and Harary 1967).
Outerplanarity example

Figure 2: An outerplanar graph and forbidden subgraphs $K_4$ and $K_{3,3}$. 
Approximation algorithms

- An approximation algorithm gives always a solution for a given computational problem.
- It is possible that this solution is not optimal!
- The performance ratio $R_A$ of an approximation algorithm $A$ is the worst case ratio of obtained solutions to the cost of optimal solution:

$$R_A = \min \frac{A(G)}{OPT(G)}, \text{ where } OPT(G) \neq 0$$

and where $G$ is any possible input graph (if $OPT(G) = 0$, then divider is set to 1).
- The ratio is always at most 1 and the algorithm produces better approximations if the ratio is closer to 1.
How to prove the performance ratio?

• First, prove that the performance ratio $R_A \leq k$, where $k \in \mathbb{R}$.
  
  -- To prove the upper bound it is sometimes possible to construct such a problem instance that the claimed upper bound is achieved. Only one such instance is enough!

• Second, prove that the performance ratio $R_A \geq l$, where $l \in \mathbb{R}$.
  
  -- This is often the hardest part. Now we have to show that the lower bound holds for all possible problem instances!

• If now $k = l$, then we have $R_A(I) = k$, otherwise $l \leq R_A \leq k$. 

Spanning tree heuristic

- One way to approximate the maximum planar subgraph problem is to produce a spanning tree of the input graph.
- Without loss of generality, we may assume that graph is connected.
- For a graph with $n$ vertices, the spanning tree contains $n - 1$ edges.
- Since a planar subgraph contains at most $3n - 6$ edges, the ratio of the edges in a spanning tree and the “best possible solution” is at most $1/3$ as $n$ tends to infinity:

$$
\lim_{n \to \infty} \frac{n - 1}{3n - 6} = \frac{1}{3}.
$$

- Since a spanning tree contains always $n - 1$ edges, the ratio is at least $1/3$. Therefore, the performance ratio of the spanning tree heuristic is $1/3$. 
Greedy algorithm

- Any algorithm that *just scans the edges in some order* and tries to add them to the planar subgraph, yields an algorithm with performance ratio $1/3$.

**input**: graph $G = (V, E)$  
**output**: maximal planar subgraph $G'$ of $G$

    let $G' = (V, E')$, where $E' = \emptyset$;
    forall edges in $E$ do
        add $e$ to $E'$;
        if $(V, E')$ is not planar
            delete $e$ from $E'$;
    od
    output $G'$;
end;
Cactus-tree heuristic

- Cactus-tree heuristic was first approximation algorithm with non-trivial performance ratio (J. Algorithms, 1998).
- Heuristic is based on searching triangles, that do not violate the planarity, from the input graph.
- A **triangular structure** is a graph whose cycles, if any, are triangles.
- A **triangular cactus** is a graph whose cycles, if any, are triangles and such that all edges belong to some cycle.
- Algorithms searched first a triangular cactus and then connects it (if cactus was unconnected) to obtain a triangular structure.
- If a (suitable) triangular cactus is searched from the input graph, it guarantees performance ratio 7/18.
- If a maximum triangular cactus is searched, it guarantees performance ratio 4/9.
Algorithm A: Cactus-tree heuristic

**Input:** A graph $G = (V, E)$

**Output:** A planar subgraph $G' = (V, E')$ of $G$

**begin (phase 1)**

set $E' = \emptyset$

**while** there is a triangle $T$ in $G$ with edges $(v_1, v_2), (v_2, v_3), (v_3, v_1)$ in $(V, E)$ such that all vertices of $T$ belong to different components in $G'$ **do**

set $E' = E' \cup \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$

**end** (components of $G' = (V, E')$ are now triangular cactusses)

**begin (phase 2)**

**while** there is an edge $(v_1, v_2) \in E$ such that $v_1$ and $v_2$ belong to different components in $G'$ **do**

set $E' = E' \cup \{(v_1, v_2)\}$

**return** $(V, E')$

**end;** ($G'$ is now a triangular structure)
Analysis of the cactus-tree heuristic

• The performance ratio of Algorithm A for the maximum planar subgraph problem is $7/18$ and it runs in linear time for bounded-degree graphs.

• The proof of the lower bound is of “medium” complexity. Proof is based on the following observation:
  A triangle was not added to $G'$ if two of its vertices were already in the same component in $G'$.

• To prove the upper bound a sample graph was given.

• If the Phase 1 of Algorithm A is modified so that it searches the maximum triangular cactus, this leads to a $4/9$ approximation algorithm which runs in $O(m^{3/2}n \log^6 n)$ time.

• This better algorithm is very complicated (to prove and to implement). No known implementations!
Modified cactus-tree heuristic

- When a triangle is found in Algorithm $A$, it always connects three vertices from different components of $G'$.

- It is easy to see that all the vertices of a triangle need not to belong to different components of $G''$ in Phase 1 of Algorithm $A$:
  - It is enough to have two vertices $v_1$ and $v_2$ joined by an edge $(v_1, v_2)$ in one component and the third vertex $v_3$ in other component forming a triangle $\{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$
  - If we want to approximate outerplanar subgraphs, we need a restriction that the edge $(v_1, v_2)$ belongs to at most two triangles at the same time.

- Algorithm $A'$ tries first to connect one vertex from another component with two new edges adjacent to vertices joined by an edge in another component,

- or uses same triangle adding strategy as the algorithm $A$. 
Modified cactus-tree heuristic

Algorithm A’: Modified cactus-tree heuristic

**Input:** A graph $G = (V, E)$

**Output:** A planar subgraph $G' = (V, E')$ of $G$

**begin** (phase 1)

set $E' = \emptyset$;

while the number of edges in $E'$ increases during the while loop do

while there is a triangle $T$ in $G$ with edges $(v_1, v_2), (v_2, v_3)$ and $(v_3, v_1)$ such that edge $(v_1, v_2)$ belongs to one triangle in $E'$ and $v_3$ to a different component in $G'$ do

set $E' = E' \cup \{(v_2, v_3), (v_3, v_1)\}$;

if there is a triangle $T$ in $G$ with edges $(v_1, v_2), (v_2, v_3)$ and $(v_3, v_1)$ such that all vertices of $T$ belong to different components in $G'$ do

set $E' = E' \cup \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$;

end (components of $G' = (V, E')$ are planar)

**begin** (phase 2)

... connection in the same way as in the algorithm A ...

**end;**
Figure 3: Illustrations of the planar subgraphs for graph cimi-g4 (in the left) found by algorithms A (middle) and A' (right). Triangles are enumerated in (some possible) found order.
Performance analysis of A’

• The performance ratio of Algorithm A for the maximum planar subgraph problem is at least 7/18.
  – There is no such a triangle $T$ in the input graph after Phase 1 that its vertices are in different components in $G'$.

• The performance ratio of Algorithm A for the maximum planar subgraph problem is at most 1/2.
  – The construction used for Algorithm A (skipped) does not work here. But we can use a grid graph ...
  – Let $G$ be a $n \times n$ grid graph. Now each side of the grid contains $n$ vertices and $G$ has in total $n^2$ vertices and $2n^2 - 2n$ edges.
  – Algorithm $A'$ finds a planar subgraph with $n^2 - 1$ edges by constructing a spanning tree of $G$.
  – The ratio between the number of edges found by Algorithm $A'$ and the number of edges in $G$ is 1/2 as $n$ tends to infinity.
Running time analysis of A’

• It is enough to notice that in Phase 1 the step where a triangle connecting two vertices from the same component and one vertex from another component takes linear time provided that the degree of the graph is bounded.

• All other operations take linear time (the property of the original algorithm).

• Suppose that there are $m$ edges in a graph whose degree is bounded by a constant $d$.

• Each time when an edge $(v_1, v_2)$ is considered, it takes at most $d^2$ time to check adjacency lists of $v_1$ and $v_2$ to find a triangle.

• Since it is enough to consider each edge only once in the inner while loop, the algorithm runs in time $O(m)$. 
The properties of Algorithm A’

- The performance ratio of Algorithm A’ for the maximum planar subgraph problem is at least $7/18$ and at most $1/2$, and it runs in linear time for bounded-degree graphs.

- Since there is a gap between the lower and upper bound of the performance ratio of Algorithm A’, the exact performance ratio is left open.

- Conjecture: The performance ratio of Algorithm A’ for the maximum planar subgraph problem is at least $4/9$. 
Experiments

- We implemented Algorithms $A$ and $A'$ in $C++$.
- Algorithms were randomizes choosing vertices and edges randomly whenever it was possible.
- We used a test graph set containing 84 graphs.
- The number of edges in test graphs varied between 21 and 30380.
- Since algorithms were randomized, we performed $25 - 100$ distinct runs for each graph.
- We compared the best found solutions of both algorithms, the worst of $A'$ against the best of $A$ and the worst of $A'$ against the optimal solution (if known, otherwise best known).
- The solution quality of $A'$ is clearly better.
- The running time for the largest test graph was less than one second for $A$ and $A'$. 
• The best found solution of $A'$ is up to 35 percentages better than the best found solution for $A$!
A’ worst vs A best

- The worst found solution of $A'$ is up to 30 percentages better than the best found solution for $A$!
A’ worst vs OPT/best known

- The worst found solution of $A'$ is never more than $6/10$ away from the optimal!
Conclusions - future work

• The exact performance ratio of the Algorithm A’ is open.
  – Ratio is at least 7/18 and at most 1/2.
  – I think it is possible to prove that the Algorithm A’ finds more edges from a graph than the size of a maximum triangular cactus is. Successful proof will verify that the ratio is at least 4/9.
  – Could the ratio be 1/2?
A worst vs OPT/best known

- Now there are instances between ratios 7/18 and 8/18!