

Abstract

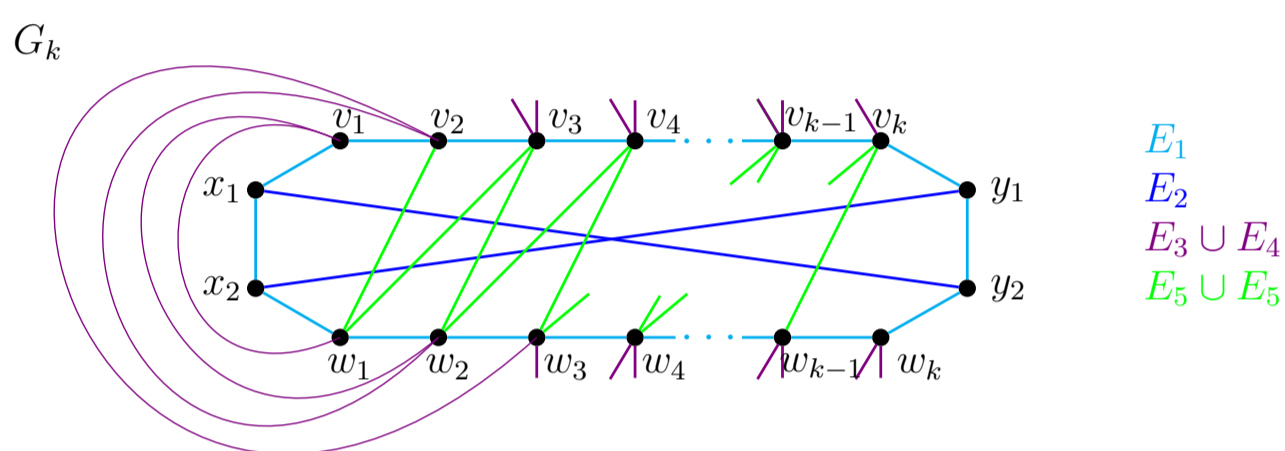
A graph is called planar if it admits a drawing in the plane such that two different edges only cross in common endpoints, i.e. a common vertex. The maximum planar subgraph problem searches for a planar subgraph with the maximum number of edges. It has many applications, e.g. in graph drawing, facility layout, circuit design and thickness approximation. Since it is an NP-hard problem, approximation algorithms returning feasible but possibly non-optimal solutions are used. To compare approximation algorithms, the approximation ratio defined as

$$\inf_{\text{all graphs } G} \frac{|E_{\text{app}}(G)|}{|E_{\text{opt}}(G)|}$$

is used, where $E_{\text{app}}(G)$ denotes the edge set of a possible output produced by the approximation algorithm and E_{opt} is the edge set of an optimal solution.

The Greedy Algorithm

The Algorithm GREEDY tries to add edges greedily to a planar subgraph maintaining its planarity. Besides a graph $G = (V, E)$, this algorithm also receives an ordered list of the edges as input. Starting with the subgraph (V, \emptyset) it tests, if the current subgraph remains planar when edge e_i is added. The edge e_i will be discarded in case it violates the planarity of the current subgraph, otherwise it will be added to the current subgraph.



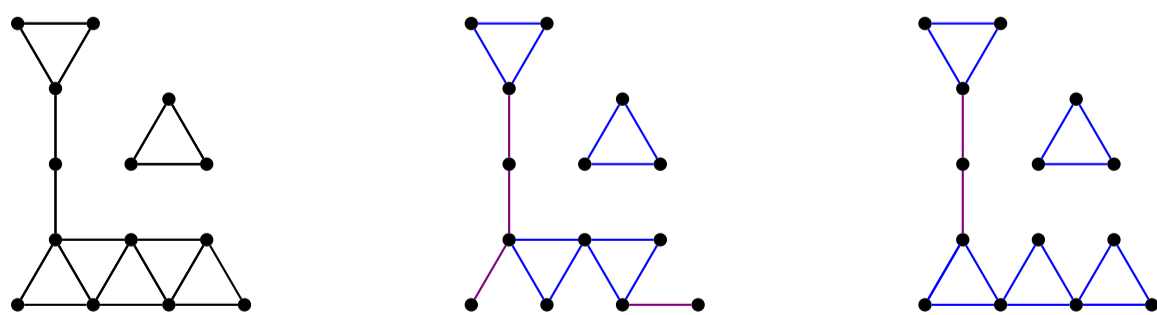
To see that the approximation ratio of the Algorithm GREEDY is at least $\frac{1}{3}$ it suffices to show that the output always contains a spanning forest of the input graph. Then using that $|E'| \leq 3|V'| - 6$ holds for planar graphs (V', E') with at least three vertices, one can easily see that the considered fraction $\frac{|E_{\text{app}}(G_k)|}{|E_{\text{opt}}(G_k)|}$ is always at least $\frac{1}{3}$. To show that $\frac{1}{3}$ is also an upper bound, the graph G_k shown in the figure above is used. The subgraph $(V, E_1 \cup E_2)$ is a maximal planar subgraph of G_k , i.e. a planar subgraph such that the addition of any other edge contained in G_k destroys the subgraph's planarity. Therefore, $(V, E_1 \cup E_2)$ is a possible output for the Algorithm GREEDY and it contains $2k + 6$ edges. On the other hand, deleting the edges in E_2 from G_k yields a maximum planar subgraph containing $6k$ edges. Therefore, the considered ratio

$$\inf \frac{|E_{\text{app}}(G_k)|}{|E_{\text{opt}}(G_k)|} \leq \frac{2k + 6}{6k} \xrightarrow{k \rightarrow \infty} \frac{1}{3}$$

approaches at most $\frac{1}{3}$ showing that the approximation ratio is at most $\frac{1}{3}$.

The Cactus Algorithms

The currently best known algorithms called cactus algorithms are based on finding a triangular cactus in the input graph. A triangular structure is a graph in which all cycles have length three. Furthermore, a triangular structure is called a triangular cactus if all edges lie in a cycle. The GREEDY CACTUS Algorithm starts with (V, \emptyset) as initial subgraph and then adds greedily the edges of triangles in the input graph, such that the current subgraph is always a triangular cactus (blue edges in the following figure). Afterwards, it adds edges that connect different components of the subgraph, such that it is always a triangular structure (purple edges in the following figure). All in all, choosing the triangles greedily an approximation ratio of $\frac{7}{18}$ is achieved. It is possible to compute a maximum triangular cactus, i.e. a subgraph that is a triangular cactus such that all other subgraphs that are triangular cacti do not contain more edges, in polynomial time. Using this maximum triangular cactus instead of the greedily constructed cactus, the Algorithm MAXIMUM CACTUS is obtained. It achieves an approximation ratio of $\frac{4}{9}$.



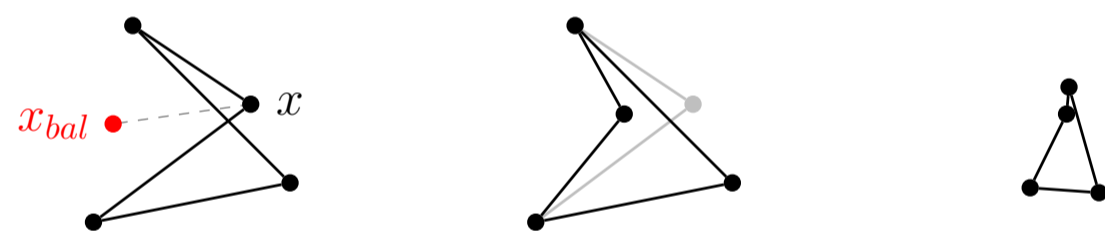
In the middle and on the right, possible outputs for Algorithm GREEDY CACTUS and Algorithm MAXIMUM CACTUS, respectively, are shown with the graph displayed on the left as input graph. However, when applied to bipartite graphs, only a spanning tree is constructed, since bipartite graphs do not contain any triangles. Restricted to bipartite graphs both versions of the Cactus Algorithm have an approximation ratio of $\frac{1}{2}$, which is also achieved by simply constructing a spanning tree of the input graph in the case of bipartite graphs. However, trying to adjust the idea of a triangular cactus such that it works for bipartite graphs does not yield better worst case results.

The Balance Point Algorithm

An intuitive approach to construct an approximation algorithm for the maximum planar subgraph problem is to first draw the input graph with as few crossings as possible and then delete as few edges as possible to obtain an embedding of a planar subgraph. However, the problem to determine this minimum number of crossings needed to embed a graph in the plane is also NP-hard, therefore an approximation algorithm is used. In order to be able to implement such an algorithm easily, all edges will be represented by straight line segments. A strategy to avoid crossings is to embed a vertex near to the balance point of its neighbors. Let $G = (V, E)$ be the input graph and denote by $x \in \mathbb{R}^2$ the coordinates of vertex $v \in V$ in the current embedding. Then the balance point of x 's neighbors is

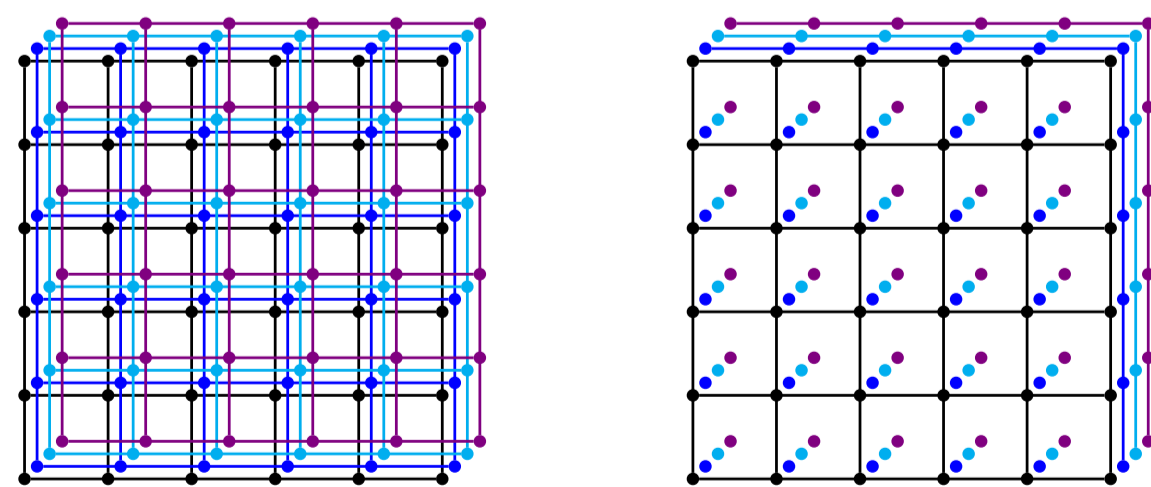
$$x_{\text{bal}} = \frac{1}{\deg(v)} \sum_{v_i \in N(v)} x_i.$$

The algorithm starts with some arbitrary initial embedding that is either given as input or random values are assigned to all coordinate vectors. Since moving one vertex changes the balance points of its neighbors, it is only moved in the direction of its balance point and not all the way. The parameter $\lambda \in (0, 1]$ specifies how far a vertex is moved in each step of the iteration, i.e. $x_{\text{new}} = (1 - \lambda)x + \lambda x_{\text{bal}}$ where x are the current coordinates of the vertex considered and x_{new} denotes its new coordinates. An example is presented in the following figure, where all points for new coordinates of x specified by the parameter λ are lying on the gray line between x and x_{bal} . Moving x with $\lambda = 0.5$ yields the drawing displayed in the middle and moving all vertices once, the drawing on the right is obtained. So all in all, the cycle got untangled by so called balance point moves and no edges need to be deleted.



At one point, usually specified by a maximum number of iterations or when the number of crossings of an embedding does not decrease anymore, the balance point moves stop and edges are deleted greedily, i.e. an edge contained in a maximum number of crossings of the current embedding is deleted first, until the embedding is planar.

However, this algorithm can achieve arbitrarily bad results. If the input graph is a $k \times k$ grid, the BALANCE POINT Algorithm will most likely untangle it. If the input graph consists of several grids, they can be embedded on top of each other as shown in the left part of the following figure. Deleting edges greedily, the right part of the following figure can be obtained. This shows that the ratio $\frac{|E_{\text{app}}(G)|}{|E_{\text{opt}}(G)|}$ can approach zero as the number of grids goes to infinity. Consequently the approximation ratio of the Algorithm BALANCE POINT is zero.



Modifying the algorithm such that different components of the input graph are embedded far away from each other does not circumvent such bad results, since one can add edges between the different grids to connect the graph used above.

References

- Călinescu, G., Fernandes, C.G., Finkler, U. and Karloff, H.: A better approximation algorithm for finding planar subgraphs. In SODA '96: Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms, pages 16–25, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.
- Yannakakis, M.: Node-and edge-deletion np-complete problems. In STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing, pages 253–264, New York, NY, USA, 1978. ACM.
- Cimikowski, R. and Coppersmith, D. The sizes of maximal planar, outerplanar, and bipartite planar subgraphs. Discrete Mathematics, 149(1-3):303 – 309, 1996.