Computer Science Journal of Moldova, vol.**15**, no.3(45), 2007

An algorithm of graph planarity testing and cross minimization

Vitalie Cotelea, Stela Pripa

Abstract

This paper presents an overview on one compartment from the graph theory, called graph planarity testing. It covers the fundamental concepts and important work in this area. Also a new approach is presented, which tests if a graph is planar in linear time O(n) and it can be used to determine the minimum crosses in a graph if it isn't planar.

1 Introduction

The planarization of a graph is a problem which was a domain of interest for many years and still is. By planarization we understand representation of a graph in such a way that any two edges, from the edge set of the graph, do not cross.

There are many areas where solving the graph planarization problem is of major importance. The research in graph planarization starts from the VLSI production to the UML diagram representations and Database Design. If we take the VLSI production – the electrical circuits are put automatically on the micro scheme made from isolating material. Because these circuits are not isolated from each other it means that they cannot cross, otherwise the micro scheme will not work properly. So the next question arises – how to put circuits in such a way that they do not cross? And if there is no solution to this, then it's necessary to determine the minimum number of crosses.

2 Known Results

The first step in the planar graphs field was made by Euler and he defined the fundamental concept in 1736:

For a simple, connected and planar graph with V-vertices and E-edges the following constraints are true [1]:

- Theorem 1: If $V \geq 3$ then $E \leq 3v - 6$
- Theorem 2: If $V \geq 3$ and there are no cycles with the length > 3, then $E \leq 2v - 4$

Another important step in the graph planarization field, belongs to the polish mathematician Kazimierz Kuratowski, which proved that a non-planar graph must contain at least a graph homeomorph to K_5 or $K_{3,3}$ (Figure 2), known as the *Kuratowski's Theorem.* The planar representation of a graph shows that a graph is planar, the presence of a Kuratowski sub graph (a sub graph homeomorphic to K_5 or $K_{3,3}$) proves that a graph isn't planar.

The graph planarization problem appeared years ago and isn't yet resolved. Further the most important results in getting a solution for this problem are presented:

- It appears in 1961 with the Auslander and Parter [2] work. They are the first who proved that planarity test has a polynomial complexity, with an execution time of $O(n^3)$ which is by far not an optimal one.
- Followed 13 years of trying to reduce the complexity. One of the simplest proposed algorithms was the algorithm proposed by Demoucron, Malgrange and Partuiset [3], but its complexity is $O(n^2)$.
    - ◇ A relatively simple algorithm, which appeared to be very useful later, was proposed by Lempel, Even and Cederbaum [4] in 1966.

- ◇ An important result was obtained by Hopcroft and Tarjan who presented a linear algorithm for planarity test [5]. This algorithm has its minuses: it's hard to understand and it's harder to implement it in linear time. Nevertheless the algorithm determines if a graph is planar, though does not find a planar representation.

- In 1976 two independent results showed that the algorithm proposed by Lempel, Even and Cederbaum, can be implemented in a linear time O(n).

  Even and Tarjan [6] proved that the crucial element for the algorithm, *st-order*, can be computed in linear time. Booth and Lueker [7] defined the PQ-trees notion with which some of the algorithm conditions can be tested in linear time. The results from [4,6,7] form a linear planarity test. But the algorithm isn't so simple because the complexity of implementing PQ-trees is high and this makes the algorithm of graph planar embedding to have an $O(n^2)$ complexity.

- After 9 years Chiba et al [8] showed the method of finding the planar embedding of a graph in a linear time using the Lempel, Even and Cederbaum algorithm.

- The graph planarity test algorithms revived when software companies started to elaborate graph libraries. In 1996 Mehlhorn and Mutzel implemented the Hopcroft and Tarjan algorithm as part of the LEDA project. They explained how can a planar embedding of a graph in linear time be obtained using this algorithm [9].

- In 1996 Di Battista and Tamassia developed an on-line planarity test algorithm [10]. It consists of a series of modifications to the graph and testing the obtained graph.

- In 1999, Boyer and Myrvold [11,12] presented a new approach on planarity testing based on depth search with corresponding

calculus.

3 Matrix Based graph planarity test algorithm:

A truly simple graph planarity test algorithm which can be explained in at most one academic hour is waiting to be found. At the moment there are two options: to forget linearity and to implement the algorithm of Demoucron et all or to use an existing package (LEDA for example) which has implemented the linear planarity test algorithm [13].

Below the new approach on testing the planarity of a graph is presented. It is based on representing the graph using the adjacency matrix and finding such vertices (v_i, v_j) that by deleting the edge e_{ij}, the graph's main structure will not change.

The adjacency matrix of a simple graph is a matrix with rows and columns labeled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are adjacent or not. [14]

For example – for the graph from the Figure 1 we will obtain the matrix from Table 1.

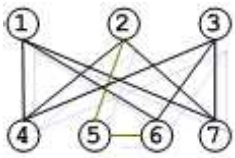

Figure 1. Graph Example

From the *Kuratowski's Theorem* it's known that a graph has a planar embedding if it doesn't contain sub graphs homeomorph to $K_{3,3}$ or K_5. The respective graphs are presented in Figure 2 and their adjacency matrix in Tables 2 and 3.

Table 1. The adjacency matrix of the graph from Figure 1

	1	2	3	4	5	6	7
1	-	0	0	1	0	1	1
2	0	-	0	1	1	0	1
3	0	0	-	1	0	1	1
4	1	1	1	-	0	0	0
5	0	1	0	0	-	1	0
6	1	0	1	0	1	-	0
7	1	1	1	0	0	0	-

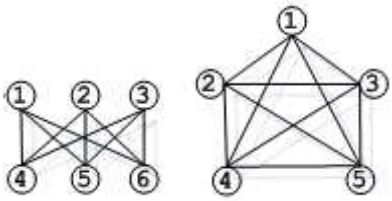

Figure 2. $K_{3,3}$ and K_5 graphical representation.

Table 2. $K_{3,3}$ adjacency matrix.

$K_{3,3}$						
	1	2	3	4	5	6
1	-	0	0	1	1	1
2	0	-	0	1	1	1
3	0	0	-	1	1	1
4	1	1	1	-	0	0
5	1	1	1	0	-	0
6	1	1	1	0	0	-

Table 3. K_5 adjacency matrix.

K_5					
	1	2	3	4	5
1	-	1	1	1	1
2	1	-	1	1	1
3	1	1	-	1	1
4	1	1	1	-	1
5	1	1	1	1	-

3.1 The Algorithm workflow:

Step 1: The adjacency matrix is computed
Step 2: Eliminate all those vertices from the graph which do not change its main structure.

Step 2.1: Find potential transformable vertex.

Each column k of the matrix will be considered as a vector V_k with ordered descending components each i where $a_{ik} = 1$ (a_{ik} is the element of the adjacency matrix). For example the vector V_6 from the Table 1 will have the following structure: V_6=[5,3,1]

From i=n downto 1

If $V_i \cap V_{Vi[j]} = \emptyset$ (where $k = Vi[j] < i$) then k is a potential transformable to i vertex.

For example: in Table 1 we have: V_6=[5,3,1] and V_5=[6,2] and $V_6 \cap V_{V6[1]} = V_6 \cap V_5 = \emptyset$. So vertex 5 is a potential transformable to vertex 6.

Step 2.2 Check if the potential transformable vertex k can be transformed in i.

Using the columns $L_j = V_j$; $j = 1, n$ compose the expanded adjacency vertex lists L_k and L_i, taking no consideration to the edge e_{ik} which links these vertices.

An expanded adjacency vertex list is the list of all vertices which can be direct accessed after potential graph transformation.

Check recursively if elements from list L_k can be transformed in at least one element ($k <> i$) of the list L_i. If doesn't exist such an element then vertex k is transformable to i.

For example: in Table 1 we have L_5=[6,2] L_6=[5,3,1]. 2 cannot be transformed in 3 or 1 because there is no connection between 2 and 3 or between 2 and 1. Vertex 5 cannot be transformed in 2 because L_2=[4,5,7], there is a connection between vertex 5 and vertex 6 and $V_6 \cap V_5 = \emptyset$. So vertex 5 is transformable to 6 and not transformable to vertex 2.

Step 2.3 Transform matrix.

If vertex k is transformable to i then $V_i = V_i \cup V_k$, forward read column k as column i.

Step 3: Test if the obtained adjacency matrix contains structures of $K_{3,3}$ *or* K_5.

Structure of $K_{3,3}$ are present if $\exists Vi \cap V_j \cap V_k >= 3$ where $i <> j <> k$.

Structure of K_5 are present if $\exists |Vi| \geq 4$ and components are dependent with each other as the K_5 structure asks.

3.2 Complexity and correctness:

Step 1: Is considered as input.

Step 2:

Step 2.1: Correctness – the vertex k cannot be transformed into vertex i if they have common edges, because in this way information is lost about the main graph structure.

Complexity – Each edge of the graph is studied, from the Euler theorem (1), they are $3n - 6$.

Step 2.2: Correctness – the vertex k cannot be transformed into vertex i if they have common edges, because in this way information is lost about the main graph structure.

Complexity – it's hard to approximate how many potential transformable vertices can be in a graph. In this case we can suppose that all $3n-6$ combinations of vertices are potential transformable. Each vertex has in medium 3-4 combinations. So for finding the answer if two vertices are transformable it will be needed maximum $4 \cdot 4 = 16$ operations to each vertex.

Step 2.3: Correctness – evident from steps 2.1 and 2.2.

Complexity – for each transformable vertex - 1 operation.

Step 3: Correctness – pure application of the Kuratowski's Theorem.

Complexity – the presence of $K_{3,3}$ or K_5 can be done simultaneously and the complexity of finding the $K_{3,3}$ is bigger because there is no dependency between horizontal and vertical values. For finding the $K_{3,3}$, each vertex column which has 3 or more edges must be checked. In the worst case there will be $2 \cdot (3n-6)$ operations.

Total Algorithm **Complexity =57n-114**

The graph must be simple and connected, otherwise the complexity will be modified and some small modifications to algorithm must be made.

4 Conclusions and Future work:

This algorithm can also be implemented:

- at determining if a graph G contains a homeomorph subgraph D. (The step 3 will consist of finding the structure of the graph D);

- at determining minimum number of crosses if the graph is not planar. The number of crosses equals to the total number of independent $K_{3,3}$ or K_5 structure found in the Step 3. (If $E \geq 3v - 6$ then the algorithm complexity will change).

The algorithm does not solve the planar embedding problem.

References

[1] http://en.wikipedia.org/wiki/Kuratowski%27s_theorem

[2] L. Auslander and S. V. Parter. *On imbedding graphs in the sphere.* Journal of Mathematics and Mechanics, Vol. 10, No. 3, pp. 517–523, 1961.

[3] G. Demoucron, Y. Malgrange, and R. Pertuiset. *Graphes planaires: reconnais sance et construction de representations planaires pologiques.* Rev. Francaise Recherche Operationnelle, vol. 30, pp. 33–47, 1964.

[4] A. Lempel, S. Even, and I. Cederbaum, *An algorithm for planarity testing of graphs,* in Proc. Theory of Graphs—Int. Symp., Rome, Italy, July 1966, pp. 215–232.

[5] J. E. Hopcroft and R. E. Tarjan. *Efficient planarity testing.* J. Association Computing Machinery, volume 21, pages 549–568, 1974.

[6] S. Even and R. E. Tarjan. *Computing an st-numbering.* Theoretical Computer Science, vol. 2, pp. 339–441, 1976.

[7] K. Booth and G. Lueker. *Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms.* Journal of Computing and System Sciences 13, pp.335–379, 1976.

[8] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. *A linear algorithm for embedding planar graphs using PQ-trees.* Journal of Computing and System Sciences 30, pp.54–76, 1985.

[9] K. Mehlhorn and P. Mutzel. *On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm.* Algorithmica 16, No. 2, pp.233–242, 1996.

[10] G. Di Battista and R. Tamassia. *On-line planarity testing.* SIAM J. Computing Vol. 25, pp.956-997, 1996.

[11] J. Boyer and W. Myrvold. *Stop minding your P's and Q's: A simplified O(n) planar embedding algorithm.* In Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, Vol. 13, pp. 140–146, 1999.

[12] Journal of Graph Algorithms and Applications http://jgaa.info/ vol. 8, no. 3, pp. 241–273, 2004

[13] CS 762: Graph-theoretic algorithms Lecture notes of a graduate course University of Waterloo Fall 2005, chapter 16, pp. 117–118.

[14] Chartrand, G. *Introductory Graph Theory.* New York: Dover, 1985, p. 218.

Vitalie Cotelea, Stela Pripa, Received October 12, 2007

Vitalie Cotelea
Academy of Economic Studies of Moldova
Phone: (+373 22) 40 28 87
E–mail: *vitalie.cotelea@gmail.com*

Stela Pripa
Academy of Economic Studies of Moldova
Phone: (+373 22) 72 71 24
E–mail: *stela.pripa@gmail.com*