A GRAPH-PLANARIZATION ALGORITHM AND ITS APPLICATION TO RANDOM GRAPHS

T. Ozawa and H. Takahashi
Department of Electrical Engineering
Faculty of Engineering, Kyoto University
Kyoto, Japan 606

Abstract. In this paper presented are a graph-planarization algorithm and the results obtained by the application of the algorithm to random graphs. The algorithm tests the subgraph of the given graph G for planarity and if the subgraph fails the test, it deletes a minimum number of edges necessary for planarization. The subgraph has one vertex at the beginning, and the number of its vertices is increased one by one until all the vertices of G are included in it. The result from the application of the algorithm to random graphs indicates that the time complexity of the algorithm is $O(n^p)$ with $p=1.4\sim1.5$ in average, where n is the number of vertices of G.

1. Introduction

Based on the planarity-testing algorithm by Lempel, Even and Cederbaum[1], a graph-planarization algorithm using PQ-trees is presented. The outline of the algorithm is as follows. First, a numbering called an st-numbering is given to the vertices of the given graph. Next, starting from the first vertex and following the st-numbering, the algorithm adds a vertex and its outgoing edges to the subgraph already obtained. Then it tests the planarity of the new subgraph, and if nonplanar, deletes a minimum number of edges necessary to obtain a planar subgraph. Once a new planar subgraph is obtained, the algorithm again adds one more vertex, and so on.

The number of edges deleted by the algorithm depends on the st-numbering and, of course, on the graph. Thus the algorithm is applied to random graphs and some statistical data are obtained. They show very interesting properties of the algorithm and random graphs, which would be useful for designing probabilistic algorithms for graphs. The properties of planarized graph can be predictied with great certainty from

the original graph.

2. Definitions

(a) Graph: The graph to be planarized is denoted by G. G has vertex set V and edge set E. The numbers of vertices and edges are denoted by n and e respectively, and the average vertex degree, by $\bar{d}$($\bar{d}$=2e/n).
(b) PQ-tree: The definitions concerning a PQ-tree are due to reference [2]. Let U be a universal set and S, a subset of U. A PQ-tree is a date-structure introduced to obtain, among all the possible sequences of the elements of U, all the sequences in which the elements of S are consecutive. In general, a PQ-tree has a tree-structure from a root to leaves. The leaves correspond to the elements of U. Reading the leaves of the tree from left to right gives an allowable sequence of the elements of U, and is called the frontier of the tree. A branching point in the tree-structure is a P-node or a Q-node, and
(1) the children of a P-node can be permuted arbitrarily, and
(2) the children of a Q-node can be reversed.

When S is given, the PQ-tree is transformed, under the restrictions (1) and (2) above, so that the elements of S are consecutive in the frontier. If the transformation is possible, the PQ-tree is called reducible. Then a new PQ-tree is constructed satisfying the condition that the elements of S be consecutive. The operation to obtain the new tree is called the reduction of the tree.

A PQ-tree is drawn on a plane with the root at the top and the leaves at the bottom. A node and its children in the tree differ in hight by one.
(c) St-numbering: Given a graph G and two vertices s and t in G. A numbering satisfying the following conditions is called an st-numbering for G. Let the number of vertex x be denoted by f(x). Conditions: (i) f(s)=1 and f(t)=n; (ii) for $x \neq s$ or t, there exist two vertices y and z such that $f(y)<f(x)<f(z)$ and y and z are adjacent to x. It is always possible to determine an st-numbering for a biconnected graph. If an st-numbering is determined, an edge in G is oriented so that it comes out from a vertex with a smaller number and goes into a vertex with a larger number.

3. Planarization Algorithm

Let G be a biconnected graph.

[Algorithm PLAN]

Step 1. Choose two vertices s and t in G, and determine an st-numbering for G. Construct PQ-tree T_1 consisting of a leaf corresponding to vertex 1 of G. Set N←1.

Step 2. For vertex N of G, construct PQ-tree T_{NN} consisting of a P-node and its children. The P-node is the root and its children are all leaves corresponding to the vertices into which the edges from vertex N go. Add T_{NN} to T_N by replacing the leaf in T_N corresponding to vertex N with the root of T_{NN}. Let T_{N+1} be the resultant PQ-tree. Set N←N+1.

Step 3. Let S be the set of all the leaves in T_N corresponding to vertex N of G. If T_N is reducible for S, then go to step 5.

Step 4. Delete a minimum number of leaves in T_N such that T_N becomes reducible.

Step 5. Reduce T_N. If N=n, stop. If N<n, go to step 2.

If PQ-trees are used to implement the planarity-testing algorithm of reference [1], the above algorithm except step 4 will be obtained.

4. Deletion of Leaves

4.1 Step 4 of PLAN

The following definitions are given first.

(a) S-leaves, S-trees, S-root and S-nodes: The leaves of S are called S-leaves. The subtree of T_N which contains all the S-leaves in its frontier, and has the minimum height possible, is called an S-tree. (The S-tree is the pertinent tree of reference [2].) The root of the S-tree is called the S-root, and a node in the S-tree having at least one S-leaf as its descendant, is called an S-node.

(b) The following four types of nodes are defined for the nodes in the S-tree.

Type B node: A node whose descendant leaves are all S-leaves.

Type W node: A node none of whose descendant leaves are S-leaves.

Type H node: A node such that the frontier of its descendant leaves has consecutive S-leaves either on the left or on the right end of the frontier. Non-S-leaves are also consecutive in the frontier.

Type A node: A node such that the frontier of its descendant leaves has consecutive S-leaves on the middle and consecutive non-S-leaves on the ends of the frontier (on both sides of the consecutive S-leaves).

A leaf is either type B or type W. The minimum number of leaves which need be deleted to make an S-node into type B, W, H and A are denoted by b, w, h and a respectively. A type B or type W node is regarded, if necessary, as a special case of a type H or type A node, when h or a is computed. The total number of descendant leaves of a node is denoted by l. We have the following proposition.

[Proposition]

A necessary and sufficient condition for a PQ-tree with given S to be reducible is that the S-root is of one of the types B, H and A.

Based on the above proposition, the step 4 of PLAN is performed in the following four substeps.

[Step 4 of PLAN]

Step 4.1 Search T_N for the S-tree starting from S-leaves and going up from children to parents. At the same time count, for each S-node, the number of S-nodes contained in its children. Let tne number be denoted by n_s.

Step 4.2 Compute b, w, h and a for each S-node by algorithm SCAN given below.

Step 4.3 When b, h and a for the S-root are obtained, the minimum of them determines the type of the S-root which makes T_N reducible with minimum deletion of edges. Once the type of the S-root is determined, the types of its descendant are sequentially determined, first its children and then its grandchildren, and so on. The nodes in T_N are first labeled B, W, H or A according to the types thus determined, and then processed as described in section 4.3.

When a new PQ-tree T_{N+1} is constructed at step 2 of PLAN, the new P-node is given a number which is larger than the number of its parent. The root of T_1 is given number 1. These numbers are utilized to obtain the S-tree efficiently.

4.2 Computation of the Minimum Number of Leaves to Be Deleted

(a) Computation of l: When T_{NN} is constructed at step 2 of PLAN, the number of leaves which the P-node has is equal to the number of edges coming out of vertex N in G. Let this number be d_N. When T_{NN} is added to T_N, the number l for the P-node is set to d_N, and the number l for each of its ancestor nodes is changed as $l \leftarrow l+d_N$.

(b) Computation of b, w, h and a: The S-tree is scanned sequentially

from the S-leaves to their parents and then to their ancestors. A queue of nodes is prepared for this scanning. A stack is also prepared for the determination of the types of nodes which takes place in the reverse order of the scanning.

[Algorithm SCAN]

1° Put S-leaves into the queue and the stack. For each S-leaf set w=1.

2° Let X be the first node of the queue. For X set b=l-w. Compute h and a for X by the method given below.

3° If X is the S-root, stop.

4° Add w for X to w for the parent of X.

5° Set $n_s \leftarrow n_s$-1 for the parent of X. If new $n_s \geq 1$, go to 2°. If n_s=0, put the parent of X at the end of the queue and the stack. Go to 2°.

For a leaf h=0 and a=0. Consider a node X which is not of the desired type. Its children are numbered from 1 to m. The numbers b, w, h and a for child i(i=1,2,..,m) of X are denoted by b_i, w_i, h_i and a_i respectively.

[Theorem 1] If X is a P-node,

$$h=\sum_{i=1}^{m} min\{w_i,b_i\}-\max_{1\leq i\leq m}(min\{w_i,b_i\}-h_i), \tag{1}$$

$$a=min\{a_I,a_{II}\} \tag{2}$$

where

$$a_I=\sum_{i=1}^{m} min\{w_i,b_i\}-\max_{1\leq i\neq j\leq m}(min\{w_i,b_i\}-h_i+min\{w_j,b_j\}-h_j), \tag{3}$$

$$a_{II}=\sum_{i=1}^{m} w_i-\max_{1\leq i\leq m}(w_i-a_i). \tag{4}$$

(Proof) In order to change X to type H with a minimum number of deleted edges it is necessary and sufficient to assign type H to one of its children which gives $\max_i(min\{w_i,b_i\}-h_i)$, and to each of the other children, type B if $b_i \leq w_i$, or type W if $b_i > w_i$. Thus eq.(1) follows. In order to change X to type A two ways are possible. The first one is to assign type H to two of its children and to each of the other children, either type B or type W. The second one is to assign type A to one of children and type W to all the other children. The minimum number of edges to be deleted in the first can be computed similarly to h, and is given by eq.(3). To obtain a minimum number of edges deleted in the second, the child with $\max_i(w_i-a_i)$ is assigned type A, and eq.(4) follows.

Obviously the smaller of the two gives a.

[Theorem 2] If X is a Q-node,

$$h=\min_{1\leq k\leq m}\left(\min\left\{\sum_{i=1}^{k-1}(w_i-b_i)-b_k+\sum_{i=1}^{m}b_i,\ \sum_{i=1}^{k-1}(b_i-w_i)-w_k+\sum_{i=1}^{m}w_i\right\}+h_k\right) \tag{5}$$

$$a=\min\{a_I, a_{II}\} \tag{6}$$

where

$$a_I=\sum_{i=1}^{m}b_i-\max_{1\leq j<k\leq m}(y_j+z_k) \tag{7}$$

$$y_j=\sum_{i=1}^{j-1}(b_i-w_i)+b_j-h_j \tag{8}$$

$$z_k=\sum_{i=k+1}^{m}(b_i-w_i)+b_k-h_k\ , \tag{9}$$

$$a_{II}=\sum_{i=1}^{m}w_i-\max_{1\leq i\leq m}(w_i-a_i). \tag{10}$$

(Proof) In order to change node X to type H, it is necessary and sufficient to assign type H to one of its children, type B(or W) to the left siblings of the child and type W(or B) to the right siblings. The number of deleted leaves is, then

$$\sum_{i=1}^{k-1}w_i+h_k+\sum_{i=k+1}^{m}b_i \quad \text{or} \quad \sum_{i=1}^{k-1}b_i+h_k+\sum_{i=k+1}^{m}w_i\ ,$$

and eq.(5) follows. Next, there are two possible ways to change X to type A. The first one is to assign type H to two of its children, type B to all the children inside of the two and type W to all the children outside of the two. Let j and k be the two children assigned type H, then the number of leaves deleted is

$$\sum_{i=1}^{j-1}w_i+h_j+\sum_{i=j+1}^{k-1}b_i+h_k+\sum_{i=k+1}^{m}w_i,$$

and eq.(7) follows. The second way is to assign type A to one of the children of X and type W to all the other. The minimum number of leaves to be deleted can be derived similarly to the case where X is a P-node.

To compute eqs.(1)(3)(4)(5) and (10) it is sufficient to scan the children of X once from child 1 to child m. To compute eq.(7) the children of X are scanned once from child 1 to child m and once from child m to child 1. An algorithm for computing a_I is shown below. The maximum

of y_i for $1 \leq i \leq j$ which is obtained by scanning from child 1 to child j, is denoted by L_j. The number of the child which gives the maximum is recorded at C_j. $L_{k-1}+z_k$ obtained by scanning from child m to child k is denoted by R_k and $max\, R_i (m \leq i \leq k)$ is denoted by M. The number of the child which gives this maximum is recorded at c.

[Algorithm QA1]

1° $L_1 \leftarrow b_1 - h_1$, $C_1 \leftarrow 1$ and $j \leftarrow 1$.

2° $j \leftarrow j+1$. If $j=m$, go to 4°.

3° $L_j \leftarrow L_{j-1} + h_{j-1} - w_{j-1} + b_j - h_j$. If $L_j > L_{j-1}$, then $C_j \leftarrow j$ and go to 2°. If $L_j \leq L_{j-1}$, then $L_j \leftarrow L_{j-1}$, $C_j \leftarrow C_{j-1}$ and go to 2°.

4° $z_m \leftarrow b_m - h_m$, $R_m \leftarrow L_{m-1} + z_m$, $M \leftarrow R_m$, $c \leftarrow m$ and $k \leftarrow m$.

5° $k \leftarrow k-1$. If $k=1$, let $a_I = (b$ of node X$) - M$, and stop. The number of the children of type H giving a_I are c and C_c.

6° $z_k \leftarrow z_{k+1} + h_{k+1} - w_{k+1} + b_k - h_k$ and $R_k \leftarrow L_{k-1} + z_k$. If $R_k > M$, then $M \leftarrow R_k$, $c \leftarrow k$ and go to 5°. If $R_k \leq M$, go to 5°.

In some case the minimum deletion of edges can be obtained by deleting an S-leaf or a non-S-leaf. In such a case the non-S-leaf is deleted, since it can be expected that the number of deleted edges will be less when algorithm PLAN proceeds to a vertex with a larger number. In our computer program the above algorithm QA1 is modified to include this operation.

4.3 Deletion Procedure

In step 4.3 of PLAN the nodes which have been put in the stack by SCAN are retrieved one by one and labeled. From the label of a node the children to be deleted can be determined. If the children deleted are leaves, they are recorded in a list, and if not, they are labeled D. If a node labeled D is not an S-node, it is put in a queue. A node with label D is taken out later from the stack or the queue, and is processed in the same way as above.

5. Reduction

The label of a node X in the S-tree and the structure of the subtree which has X as the root, can be classified into several patterns, and thus the reduction procedure for each of the patterns can be formulated.

It is only necessary to find a pattern for X and then apply the formula for the pattern to modify the S-tree. This operation corresponds to the templete matching of reference [2], but since the final step in this reduction is to replace the consecutive S-leaves with one leaf, the reduction formulae can be simpler than those in reference [2]. For the simplified formulae see reference [3].

6. Computational Complexity

The most time-consuming procedures in the steps obtaining the S-tree for a vertex in G and reducing it, are that to determine the types of S-nodes by computing the numbers b, w, h and a, and that to apply the reduction formulae. These procedures are repeated for S-nodes. The number of S-nodes varies depending on G, but its upper bound can be given as follows. The PQ-tree T_N represents a subgraph with vertices 1, 2,.., and N of G. The leaves and Q-nodes of T_N correspond to a part of edges and biconnected components of the subgraph respectively. Each of these components contains at least one edge which does not correspond to a leaf. Thus the sum of the numbers of leaves and Q-nodes in T_N is no more than the number of edges in the subgraph. The number of P-nodes in T_N is no more than the number of vertices in the subgraph. Thus the total number of nodes including leaves in T_N is no more than the sum of the numbers of edges and vertices of the subgraph. This sum is no more than e+n.

The steps for the S-tree are repeated at most n times. Thus the time complexity of PLAN is no more than $O(n(e+n))$.

7. Application of PLAN to Random Graphs

Algorithm PLAN was computer programmed and applied to random graphs. The number n of vertices of generated graphs ranges from 30 to 150, and the average vertex degree $\bar{d}$=6, 8, 10 or 12. Note $\bar{d}<<n$.

7.1 General Description

PQ-tree T_N for large N usually contains one large Q-node, which represents a large biconnected component(a component with many edges and vertices). The reason for getting such a large biconnected component is

conjectured as follows. Even if there exist more than one large biconnected components they will be connected together after a small increase of N, because the probability for existence of edges between the vertices in these components and vertex N, is large. S-leaves are, in general, scattered around under the Q-node, and thus most of them are deleted leaving only one, two or three of them. (Note that the average indegree of a planar graph is less than or equal to three.) A non-S-leaves is rarely deleted.

7.2 Dependence of Deleted Edges on St-numberings

The number of deleted edges, denoted by e_d, for planarization depends on st-numberings. An example of the frequency distribution of e_d obtained from 200 st-numberings is shown in Fig. 1. This example and other 15 examples for n=50, 80, 100 and 120 indicate that e_d is rather constant for various st-numberings, that is, the standard deviation of e_d is far smaller than the average of e_d. It is also found that the standard deviation and the average are approximately the same for a small number and large number of trial st-numberings. Fig. 2 shows an example. This means they can be predicted with rather small number of trial st-numberings, and then the lower limit of e_d which can be obtained by PLAN, is

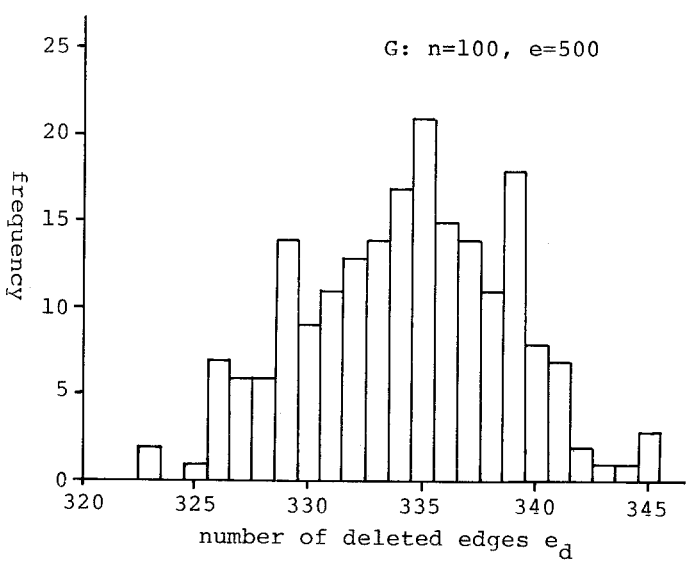

Fig. 1 Frequency distribution of the number of deleted edges e_d for various st-numberings. The average is 332.5, and the standard deviation is 4.3.

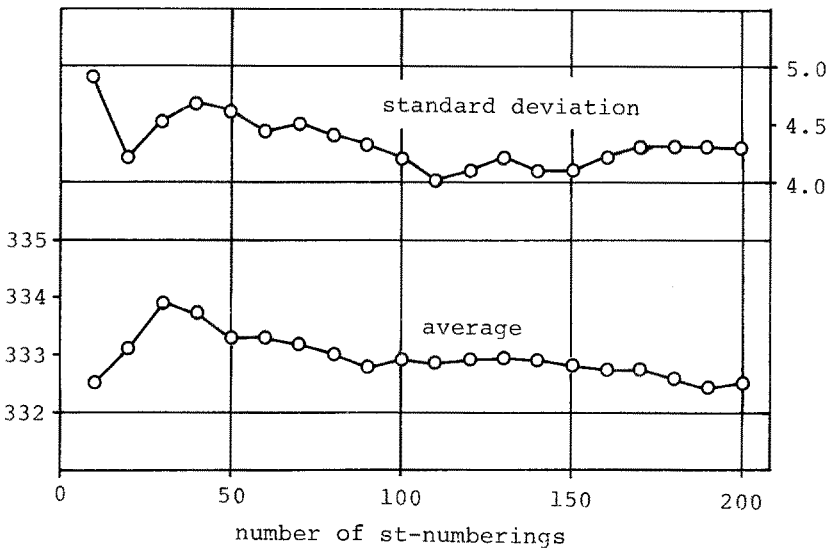

Fig. 2 Variation of the average and the standard deviation of e_d for the number of st-numberings.

estimated from them.

Table 1 shows the ranges(in upper cases) and the averages(in lower cases) of e_d taken over 10 different st-numberings for a graph with specified n and $\bar{d}$.

Table 1 The ranges of the averages of e_d.

$\bar{d}$ \ n	30	50	70	100	120	150
6	32 ∿ 43	71 ∿ 79	102∿115	158∿162	207∿210	
	38	76	109	160	209	
8	54 ∿ 62	106∿116	156∿166	241∿251	289∿305	383∿394
	59	113	162	245	298	388
10	77 ∿ 87	151∿160	217∿228	325∿338	402∿413	514∿531
	83	155	223	334	410	524
12	106∿112	191∿203	278∿290	419∿433	511∿523	658∿668
	109	196	286	425	517	663

7.3 Dependence of e_d on graphs

The average of e_d for 10 st-numberings is obtained for each graph with specified n and $\bar{d}$, and then the range and the average of the averages for 10 different graphs with the same n and $\bar{d}$ are obtained. The results for various n and $\bar{d}$ are shown in Table 2.

Table 2 Variation of e_d for 10 graphs.

$\bar{d}$ \ n	30	50	70	100	120	150
8	57∿60	106∿113	162∿166	241∿249	296∿305	383∿391
	58	111	164	245	300	387
10	82∿84	149∿155	222∿227	330∿337	405∿412	518∿527
	83	151	224	333	408	523
12	108∿110	193∿197	283∿287	420∿428	512∿519	655∿664
	109	195	286	424	517	660

This table indicates that e_d is rather constant for various graphs, if n and $\bar{d}$ are the same.

The average of the average vertex degrees, denoted by $\bar{d}_p$, of 10 planarized graphs is plotted in Fig. 3 for various n. The parameters in the figure are the average vertex degree $\bar{d}$ of the original graphs and the ratio $\beta=\bar{d}/(n-1)$, which is the ratio of the number of edges of G to that of the complete graph with the same number of vertices. (It is expected that a maximal planar graph is obtained from a complete graph.) From this figure the number of edges of a planarized graph can be predicted with great certainty from the original graph.

7.4 Computation Time

The average computation time required for planarization vs the number of vertices of the graphs is shown in Fig. 4. Ten different graphs with specified n and $\bar{d}$, and ten different st-numberings for each of the graphs were tried. (Thus the average is taken over 100 applications of the algorithm for each pair of n and $\bar{d}$.) From this figure it is seen that the computation time is of $O(n^p)$ with p=1.41∿1.44. The computer used is FACOM M200 of the Data Processing Center at Kyoto University, Kyoto, Japan.

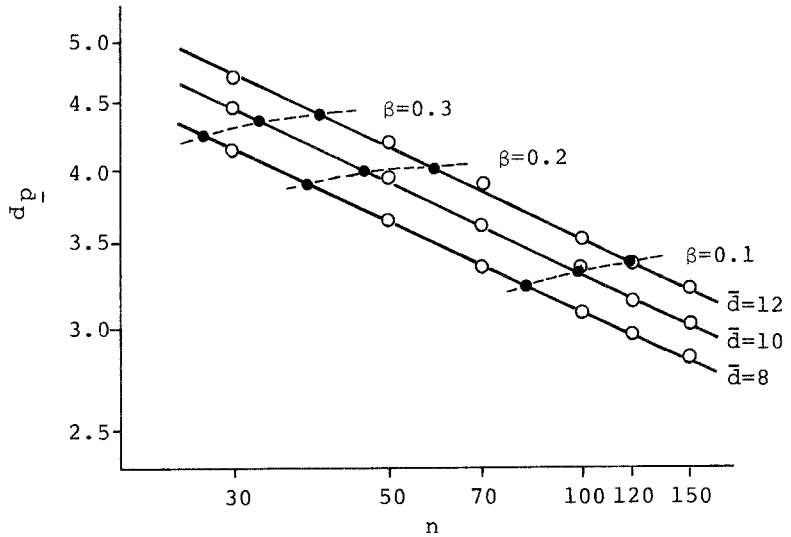

Fig. 3 Average vertex-degree of planarized graphs vs number of vertices of the original graphs.

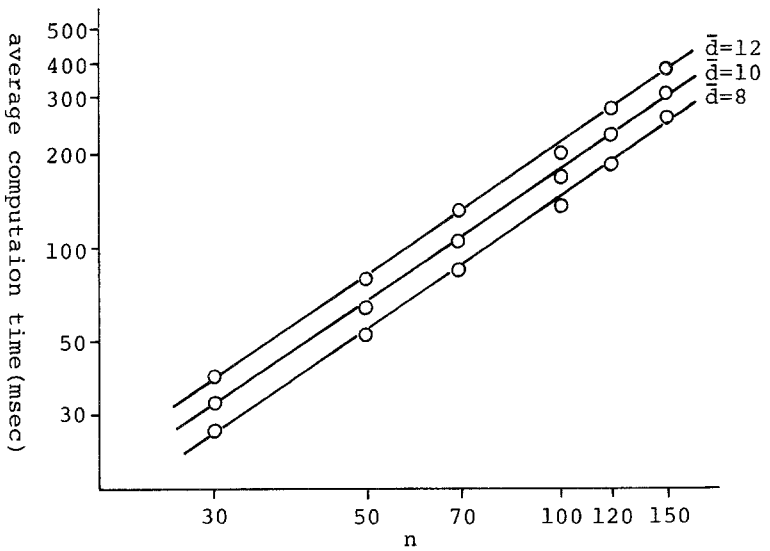

Fig. 4 Average computation time for planarization vs number of vertices of the original graphs.

8. Concluding Remarks

The planarization algorithm using PQ-trees has an advantage, over that using path embedding[4], that the mapping of planarized graphs can be easily obtained from the PQ-trees. From the results as shown in Fig. 1, it is expected that the algorithm gives e_d which is very close to the minimum required for planarization. If so, PLAN is an efficient approximation algorithm for the maximum planarization problem which has been proved to be NP-complete[5]. The computational results obtained here would be useful for studying the properties of random graphs[6].

Acknowledgements This work was supported in part by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant: Cooperative Reseach (A) 435013(1979).

Refernces

[1] Lempel, A., Even, S. and Cederbaum, I. "An algorithm for planarity testing of graphs," Theory of Graphs, International Symposium, Rome, July 1966, Rosenstiel, P. edit., pp.215-232, Gordon & Breach, N. Y., 1967.

[2] Booth, K. S., and Lueker, G. S. "Testing for the consecutive ones property, interval graphs and graph planarity using PQ-trees," J. Computer and Syst. Scie., vol.13, pp.335-379, 1976.

[3] Ozawa, T and Takahashi, H. "An algorithm for planarization of graphs using PQ-trees," Trans. Information Processing Society of Japan, vol.22, pp.9-15, 1981; and also Tech. Report, Inst. Electronics and Communication Engineers of Japan, Circuits and Systems, CAS79-150, Jan. 1980.

[4] Chiba, T., Nishioka, I. and Shirakawa, I. "An algorithm of maximal planarization of graphs," 1979 International Symposium on Circuits and Systems Proc. pp.649-653, 1979.

[5] Garey, M. and Johnson, D.:"Computers and Intractability," W. H. Freeman and Co., Reading, England, 1979.

[6] Ozawa, T. and Nishizeki, T. "Properties of certain types of random graphs," 1979 International Symposium on Circuits and Systems Proc., pp.88-91, 1979.