社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－SLDM－113　(12)
2004／1／23

反転禁止部分グラフを含む平面グラフ抽出法の効率化

木下　敏行[†]　　高藤　大介[†]　　渡邊　敏正[†]

† 広島大学大学院 工学研究科 情報工学専攻
〒739-8527 東広島市鏡山一丁目 4-1
(電話) 0824-24-7666 (高藤), -7662 (渡邊)
(ファクシミリ) 0824-22-7028
E-mail: †{lord,daisuke,watanabe}@infonets.hiroshima-u.ac.jp

あらまし　反転禁止部分グラフを有する平面グラフを抽出する発見的解法は現在までにいくつか提案されているが，数万頂点という実用規模のグラフモデルに対しては計算時間・使用メモリ量の面で実用に供する手法は未提案である．本稿では，このような規模のグラフモデルに対して上記問題を実用的計算時間内で解くことができる並列解法を 3 つ提案し，これらと既存の 4 つの直列解法の性能を計算機実験により比較評価する．
キーワード　平面グラフ抽出, 反転禁止，並列アルゴリズム，計算時間

Efficient Extraction of a Planar Graph with Subgraphs whose Turning Over is Forbidden

Toshiyuki KINOSHITA[†], Daisuke TAKAFUJI[†], and Toshimasa WATANABE[†]

† Graduate School of Engineering, Hiroshima University
1-4-1, Kagamiyama, Higashi-Hiroshima, 739-8527 Japan
Phone : +81-824-24-7666(Takafuji), -7662 (Watanabe)
Facsimile : +81-824-22-7028
E-mail: †{lord,daisuke,watanabe}@infonets.hiroshima-u.ac.jp

Abstract　Although several algorithms for extraction of a planar graph with subgraphs whose turning over is forbidden has been proposed, it does not seem that there exists any algorithm that can be used for large graphs appearing in practical situation. In this paper we propose three parallel algorithms that extract such planar subgraphs in realistic computation time. Performance of these three algorithms as well as four existing sequential algorithms is evaluated through experimental results.
Key words　Planar graph extraction, turn-forbiddance, parallel algorithms, computation time

1 Introduction

1.1 Definitions

The problem of extracting a maximum spanning planar subgraph is defined as follows: "Given a graph $G = (V, E)$, find an edge set $E' \subseteqq E$ with the maximum cardinality among all edge sets $E'' \subseteqq E$ such that $G' = (V, E'')$ is a spanning planar subgraph of G".

We call an algorithm for extracting such a spanning planar subgraph $G' = (V, E')$ a *planarization algorithm*. Consider any planar graph $G_p = (V, E_p)$ with directed cycles $C_i (i = 0, \cdots, k;\ k \geq 0)$ which must be embedded as specified (that is, each cycle C_i is forbidden to be turned over). Since if G_p has no such directed cycles then the problem has been conventionally considered. Hence we assume $k \geq 1$ in the following. Let $\widetilde{G_p}$ denote a plane embedding of G_p. If all C_i are embedded as specified in $\widetilde{G_p}$, $\widetilde{G_p}$ is called a plane embedding (of G_p) under "*forbiddance of turning over*". Given a graph $G = (V, E)$, a turn-forbidden planarization algorithm is an algorithm to extract a spanning planar subgraph $G_p = (V, E_p)$, with $E_p \subseteqq E$, such that $\widetilde{G_p}$ is a plane embedding under forbiddance of turning over. In order to realize a turn-forbidden planarization algorithm, we represent each specified cycle as a clockwise directed cycle, and any operation during the algorithm maintains clockwise directedness of these cycles.

For a set $S \subset V$ of a graph $G = (V, E)$, let $G[S]$ denote the graph (S, E_S), where $E_S = \{e = (u, v) \in E \mid u, v \in S\}$. $G[S]$ is called *the subgraph induced by* S of G. V or E is sometimes represented as $V(G)$ or $E(G)$, respectively. For any two vertex sets $S_i \subseteqq V$ $(i = 1, 2)$, $K(S_1, S_2; G) = \{(u_1, u_2) \in E \mid u_1 \in S_1$ and $u_2 \in S_2\}$.

1.2 Motivation

For designing printed-wiring-boards or VLSI, we often represent a given circuit as a graph model: for example, a graph model in which a path or a directed cycle represents how pins of a given element are located, and a spanning tree does a connection requirement among pins. Generally speaking, most elements and some modules have a side to be faced to a board in actual mounting, and they cannot be placed upside down. We call such an element as a *one-sided* element. Designing layout of each layer of single- or multi-layered boards requires extracting a spanning planar subgraph of a given graph model, where one-sided elements have to be handled.

If we represent each one-sided element as a clockwise directed cycle and apply a turn-forbidden planarization algorithm, then we can find planar layout in which all one-sided elements are placed as specified. Turn-forbidden planarization algorithms have great importance practically.

1.3 Known Results

The problem of extracting a maximum spanning planar subgraph problem is NP-hard [1] in general. It has been well investigated and many algorithms have ever been proposed [2] ~ [12]. Unfortunately however, any algorithm in [2] ~ [6], [10] is unlikely to be useful in such practical situations, while those in [7] ~ [9], [11], [12] can extract a spanning planar subgraph under the forbiddance of turning over. Turn-forbidden planarization algorithms are useful not only in the field of designing layout of printed-wiring-boards having one-sided elements but in extracting a spanning planar subgraph from a given graph that is too huge to be handled without reduction of its size. Algorithms for designing printed-wiring-boards or a VLSI have been proposed in [7], [9], [11], [13]. The one in [13] is based on a finding maximum-weight face: a linear time algorithm for finding a maximum-weight face of a given planar graph G_p has been proposed in [14] which also gives a linear time algorithm for finding a planar embedding $\widetilde{G_p'}$ of G_p such that the infinite face of $\widetilde{G_p'}$ is a maximum-weight face of G_p. An algorithm for finding a maximum-weight face is also proposed in [15].

Table 1 summarizes conventional planarization algorithms. The value PR in Tables 1 and 2 is defined by $PR = \min\{C'(G)/C(G) \mid \text{any graph } G\}$, where $C(G)$ or $C'(G)$ denotes the number of edges in any maximum spanning planar subgraph of G or of those in a spanning one extracted from G by each algorithm, respectively.

Table 1 Summary of conventional planarization algorithms (without forbiddance of turning over of subgraphs), where "—" denotes that PR is not known.

algorithms	computation time	PR
Edge_Embedding [2]	$O(\lvert E\rvert \log \lvert V\rvert)$	1 / 3
Traiangulation [3]	$O(\lvert V\rvert^3)$	7 / 18
Traiangulation [3]	$O(\lvert E\rvert^{\frac{3}{2}} \lvert V\rvert \log^6 \lvert V\rvert)$	2 / 5
Path_Embedding [4]	$O(\lvert V\rvert \lvert E\rvert)$	1 / 3
Cycle_Packing [6]	$O(\lvert V\rvert \lvert E\rvert^2)$	—
Incremental [5]	$O(\lvert V\rvert + \lvert E\rvert)$	1 / 3
Vertex_Addition [16]	$O(\lvert V\rvert^2)$	—

1.4 Purpose and Main Results

First, in this paper, we propose efficient parallel turn-forbidden planarization algorithms **PDD**, **PDR** and **PDMC**, all of which are heuristic ones. We experimentally compare performance of the three proposed algorithms and four known sequential algorithms *PLAN-PWB2*, *PLAN-MWW2*, *PLAN-DIVIDE* and *PLAN-DIVIDE2* .

Table 2 summarizes the three proposed algorithms and other known ones, all of which are turn-forbidden planarization ones.

Table 2 Summary of the three proposed algorithms and other known ones, all of which are turn-forbidden planarization ones, where "—" denotes that the value is not known.

algorithms	computation time	PR
PDD	—	—
PDR	—	—
PDMC	—	—
PLAN-DIVIDE [12]	$O(\lvert V\rvert(\lvert V\rvert + \lvert E\rvert))$	—
PLAN-DIVIDE2 [17]	—	—
PLAN-PWB2 [17]	$O(\lvert V\rvert(\lvert V\rvert + \lvert E\rvert))$	—
PLAN-MWW2 [17]	—	—
PLAN-PAA [8]	$O(\lvert V\rvert \lvert E\rvert)$	—
PLAN-MNC [9]	—	—
PLAN-MIS [11]	—	—

Experimental results for 130 randomly generated graphs having directed circuits show that **PDD** and **PDR** have extracted a spanning planar subgraph quickly, while the others have failed. This shows their usefulness in extracting a spanning planar subgraph of a given huge graph under forbiddance of turning over.

2 An Algorithm *PLAN-DIVIDE* [12]

In this section, we explain a heuristic turn-forbidden planarization algorithm *PLAN-DIVIDE* [12]. It is outlined in the following. Let *max_edge* be the maximum cardinality of an edge set that can be handled simultaneously by any existing planarization algorithm. The purpose of *PLAN-DIVIDE*[12] is to find hierarchically a spanning planar subgraph of a given huge graph $G = (V, E)$ containing a family of directed cycles $\mathcal{K} = \{C_1, \dots, C_k\}$ $(k \geqq 1)$. First, by utilizing a breath-first search BFS, *PLAN-DIVIDE* divides G with $|E| > max\_edge$ into some smallen graphs $G_i = (V_i, E_i)$ with $|E_i| \leqq max\_edge$, $i = 1, \cdots, d$ for some $d \geqq 1$, such that the vertex set $V(C_j) \subseteqq \mathcal{K}$ is contained in some G_i and such that $V(C_i) \cap V(C_j) = \emptyset (i \neq j)$. Then, for any $i = 1, \dots, d$, each spanning planar subgraph H_i and its plane embedding $\widetilde{H_i}$ of G_i, in which every directed cycles are drawn clockwise, is obtained by applying *PLAN-PWB2*[17].

Let $\widetilde{H_i}'$ $(1 \leq i \leq d)$ be a plane embedding such that the maximum weighted face of each $\widetilde{H_i}$ is an outer face. Second, represent the contour of the outer face of each $\widetilde{H_i}'$ as a clockwise directed cycle C_i'. Let $E_{ij} \subseteqq E$ be the set of edges connecting vertices of $V(C_i')$ and those of $V(C_j')$ for any pair $i, j \in \{1, \dots, d\}$. Let $E_c = \bigcup_{i,j=1(i \neq j)}^{d} E_{ij}$, let $V_{red} = \bigcup_{i=1}^{d} V(C_i')$, $E_{red} = E_C \cup \left(\bigcup_{i=1}^{d} E(C_i')\right)$ and $G_{red} = (V_{red}, E_{red})$. If $|E_{red}| \leqq max\_edge$, then we extract a set F_C of planar edges from E_C by applying *PLAN-PWB2* to G_{red}. If $|E_{red}| > max\_edge$, put $G \leftarrow G_{red}$, and repeat above hierarchical planarization steps recursively. After some iteration, we find G_{red} with $|E_{red}| \leqq max\_edge$ and can extract planar edges from E_C, and we obtain a planar graph $H = (V_H, E_H)$ $V_H = V$ and E_H that corresponds to those edges appeared in $E(H_i)$ or in F_C in any hierarchically repeated step.

The details of *PLAN-DIVIDE* are omitted: see [12].

3 The proposed algorithms

In this section, we propose three turn-forbidden planarization algorithms **PDD**, **PDR** and **PDMC**, all of which are parallel and heuristic ones. **PDR** is a parallelized version of the known sequential algorithm *PLAN-DIVIDE*[12]. If $G_{red} > max\_edge$ at the end of the first repetition then **PDD** finds a spanning tree T_{red} of G_{red} by means of a depth-first search(DFS), $F_C \leftarrow E(T_{red})$, and halts. Partitioning into subgraphs in **PDD** and **PDR** is done by means of BFS, while **PDMC** utilizes a minimum cut algorithm.

3.1 Algorithm PDD

Suppose that an integer $lb \leq max\_edge$ is given and that a class of processors $PE = \{PE_1, \cdots, PE_M\} (M \geqq 1)$ are available, where PE_1 is called the root. **PDD** partitions a given graph G into disjoint subgraphs $H_{S_1}, \cdots, H_{S_d}$ by using BFS with $lb \leq |E(H_{S_i})| \leq max\_edge$. **PDD** assigns each H_{S_i} to some processor PE_j. For simplicity, let us assume that H_{S_i} is assigned to PE_i, $1 \leqq i \leqq d$, where PE_1 is the root. Then *PLAN-PWB2* is executed in all PE_i, $i = 1, \cdots, d$, in parallel. **PDD** replaces each H_{S_i} with a directed cycle C_{S_i}' of G.

<u>**PDD**</u>

(Input) A graph $G = (V, E)$ with $\mathcal{K} = \{C_1, \cdots, C_k\} (k \geqq 1)$, an integer lb, and a class of processors $PE = \{PE_1, \cdots, PE_M\}$ where PE_1 is called the root.

(Output) An edge set $E' \subseteqq E$ such that $G' = (V, E')$ is planar under forbiddance of turning over.

Step 1. $\mathcal{K}_V \leftarrow \emptyset$; $E' \leftarrow \emptyset$; $H \leftarrow G$; $i \leftarrow 0$;

Step 2. If $|E(H)| \geqq lb$ then repeat the following in PE_1;

Find a vertex set $S_i \subseteqq V(H)$ by applying procedure *Find_Vertex_Set2* to H, $H \leftarrow H - H[S_i]$, and $i \leftarrow i + 1$.

Step 3. If $i = 0$ then goto Step 4. Let $G_i \leftarrow H[S_i]$, $i = 1, \cdots, d$, be the subgraphs found in Step 2 and let $\mathcal{K}_{S_i} \subseteqq \mathcal{K}$ be the class of cycles C_i contained in each G_i. Then, distribute G_i and $\mathcal{K}_{S_i}$ to PE_i, $i = 1, \cdots, d$, where we assume $d \leq M$. For each i, $1 \leqq i \leqq d$, execute the following (1)–(6) in $PE_1, \cdots, PE_M$ in parallel or in PE_1;

(1) $H_{S_i} \leftarrow H[S_i]$;

(2) Extract a spanning planar subgraph $H_{S_i}' = (S_i, E_{S_i}')$ of H_{S_i} by means of *PLAN-PWB2*, $E' \leftarrow E' \cup E_{S_i}'$; $\mathcal{K} \leftarrow \mathcal{K} - \mathcal{K}_{S_i}$ in PE_1;

(3) Calculate a vertex weight $w(v)$ $(v \in S_i)$ defined by $w(v) = |K(\{v\}, V(H) - S_i; H)|$;

(4) Find a maximum weight face f_{max} of H_{S_i}' with respect to the weight $w(v)$, $v \subseteqq S_i$, and let $\widetilde{H_{S_i}}$ be a plane embedding such that f_{max} is the infinite face;

(5) Replace the outer face f_{max} of $\widetilde{H_{S_i}}$ with a cycle C_{S_i}' by executing *Replace_Cycles* and $\mathcal{K}_V \leftarrow \mathcal{K}_V \cup \{C_{S_i}'\}$;

(6) (In PE_1) $E(H) \leftarrow (E(H) - E(H_{S_i}) - K(S_i - V(C_{S_i}'), V(H) - S_i; H)) \cup E(C_{S_i}')$, $V(H) \leftarrow (V(H) - S_i) \cup V(C_{S_i}')$, $\mathcal{K} \leftarrow \mathcal{K} \cup \{C_{S_i}'\}$;

Step 4. If $|E(H)| \leq max\_edge$ then extract a spanning planar subgraph H' of H by using *PLAN-PWB2* else find a DFS tree H by apply *PLAN-DFS* to H in PE_1.

Step 5. $E' \leftarrow E' \cup E(H') - \bigcup_{C' \in \mathcal{K}_V} E(C')$(in PE_1).

<u>Procedure *PLAN-DFS*</u>

(Input) A graph $G = (V, E)$ with $\mathcal{K} = \{C_1, \cdots, C_k\} (k \geqq 1)$.

(Output) An edge set $E' \subseteqq E$ such that $G' = (V, E')$ is planar.

Step 1. Reduce each C_i to an individual vertex for $i = 1, \cdots, k$ and let G_k denote the resulting graph;

Step 2. Find a DFS tree $T_{\mathcal{K}}$ of G_k by means of a depth-first search and $E' \leftarrow E(T_{\mathcal{K}})$;

<u>Procedure *Find_Vertex_Set2*</u>

Given a graph $G = (V, E)$ and a set of directed cycles $\mathcal{K}$ if $|E| \leq lb$ then return else *Find_Vertex_Set2* finds a vertex set $S \subseteqq V$ satisfied the following (i) and (ii): (i) $lb \leqq |E_S| \leq max\_edge$, where $G[S] = (S, E_S)$; (ii) For any $C' \in \mathcal{K}$, $V(C') \subseteqq S$ or $V(C') \cap S = \emptyset$.

Procedure *Replace_Cycles*

Procedure *Replace_Cycles* replaces each $\widetilde{H_{S_i}}$ with a directed cycle C'_{S_i} consisting of those vertices in the contour of the infinite face f_{max} of $G[S]''$.

Step 1. For any $v \subseteq V(f_{max})$, $vst(v) \leftarrow 0$; Let $V(f_{max})$ be the vertex set of the contour of the infinite face f_{max} of $G[S]''$.

Step 2. $i \leftarrow 1$.

Step 3. Let v_0 be an arbitrary vertex of $V(f_{max})$. For any $v \in V(f_{max})$ appearing in clockwise order from v_0, if $w(v) > 0$ then $vst(v) \leftarrow i$ and $i \leftarrow i + 1$.

Step 4. Let $V'_f = \{v \in V(f_{max}) \mid w(v) > 0\}$ and $n = |V'_f|$. $E(C_S)' = \{\langle v_1, v_2\rangle \mid vst(v_1) + 1 = vst(v_2) = j,\ 2 \leq j \leq n\} \cup \{\langle u_1, u_2\rangle \mid vst(u_1) = n,\ vst(u_2) = 1\}$, where $\langle v_i, v_j\rangle$ is a directed edge from v_i to v_j.

(The details of these procedures are omitted: see [12].)

3.2 Algorithm PDR

PDR executes the following Step 4' instead of Step 4 of **PDD**.

Step 4'. If $|E(H)| \leq max\_edge$ then extract a spanning planar subgraph H' of H by applying *PLAN-PWB2* to H else apply **PDR** to H recursively in PE_1.

3.3 Algorithm PDMC

PDMC repeatably partitions a graph to k subgraphs by means of a minimum cut to minimize the number of edges connecting these subgraph.

Instead of Step 4' of **PDR**, **PDMC** executes the following steps Step4'-1 and Step 4'-2.

Step 4'-1 shrink each C_i into individual and let G_k be the resulting graph. For each vertex v of G_k, if v corresponds to some cycle C then we assign v a weight equal to $|V(C)|$ else v has a weight equal to 1. If multiple edges are seated for any pair u, v of vertices of G_k then we replace them by a single edge (u, v) with a weight equal to the multiplicity.

Step 4'-2 by applying any minimum cut algorithm to G_k, partition G into k subgraphs G_i;

4 Experimental Results

4.1 Implementation

We have implemented all the algorithms on a personal computer (CPU: Pentium IV/1.7GHz, OS: Free BSD 4.5-R) with the C programming code.

4.2 Input data

5 input graphs are provided for each pair of $|V|$ and $|P|$, where $|P|$ is the number of parts that can be represented as directed cycles or paths. Graphs and circuits are generated by means of random numbers. We has set $max\_edge = 20000$ for *PLAN-DIVIDE* and *PLAN-DIVIDE2* in Tables 4 and 6 and $max\_edge = 5000$ and $lb = 3000$ in other tables.

4.3 Results and Observation

PLAN-DIVIDE2 and *PLAN-MWW2* failed to extract planar graphs in Data2 , Data3 and Data4. In Tables 4 through 13, figures in bold face denote the best solution. Observations on these results are summarized as follows.

(i) *PLAN-PWB2* failed to extract spanning planar subgraphs from graphs having more than 200,000 edges because of memory overflow, while *PLAN-DIVIDE* succeeded. The CPU time of *PLAN-DIVIDE* seems to be independent of increase in $|E|$, while this is not the case when both $|V|$ and $|E|$ become large. On the other hand, **PDD** and **PDR** do not depend on increase in $|V|$ or $|E|$.

(ii) Since solutions by **PDMC** are worse than those by **PDR**, our experimental results do not show usefulness of minimum cuts in partitioning into subgraphs.

(iii) Experimental results on Data1 show that dividing an input graph into many small graphs may decrease the CPU time of **PDR**. It is shown that $|E_p|$ by **PDR** is 95.04% of that

Table 3 Summary of input data

Data	Type	Size
Data1	General	$\|V\| \subseteq \{2000, 5000, 10000\}$ $\|V(\mathcal{K})\| \leq \frac{\|V\|}{2}$ (*1)
Data2	Circuit	$\|P\| \subseteq \{2000, 5000, 10000\}$; (*2) $A = 40,\ B = 30,\ C = 20,\ D = 10$ $5804 \leq \|V\| \leq 29137$; (*3) $14620 \leq \|E\| \leq 73266$
Data3	Circuit	$\|P\| \subseteq \{2000, 5000, 10000\}$ $A = B = C = D = 25$ $8925 \leq \|V\| \leq 44800$; $23234 \leq \|E\| \leq 116392$
Data4	Circuit	$\|P\| \subseteq \{2000, 5000, 10000\}$ $A = 10,\ B = 20,\ C = 30,\ D = 40$ $12046 \leq \|V\| \leq 60517$; $31840 \leq \|E\| \leq 159502$

(*1) $|V(\mathcal{K})| = \bigcup_{i=1}^{k} |V(C_i)|$.

(*2) $|P|$: the number of parts(the number of cycles).

(*3) $(|P|/100) \times A(B, C, D$, respectively$)$ = No. of 1- (2-, 3-, 10-) terminal parts.

Table 4 Comparison of average $|E_p|$ when known algorithms are applied to Data1

$\|V\|$	$\|E\|$	(DIV)	(MWW2)	(DIV2)	(PWB2)
2000	6000	2176	**3075**	2724	2176
2000	10000	2285.2	2301	**2752**	2285.2
2000	20000	2548.8	—	**2717**	2548.8
2000	60000	3102.4	—	**3788.2**	3227.2
2000	100000	3316.4	—	**3920.2**	3536
2000	200000	3856	—	**4130**	3983
5000	15000	**5231.2**	—	—	**5231.2**
5000	25000	5390.4	—	—	**5421.2**
5000	50000	5623.6	—	—	**5766.8**
5000	150000	6586	—	**9307**	7114
5000	250000	7203	—	**9429**	—
5000	500000	7971	—	**9516**	—
10000	30000	10284.5	—	—	**10303.4**
10000	50000	10432.5	—	—	**10487**
10000	100000	—	—	—	**11058.2**
10000	300000	**11986**	—	—	—
10000	500000	**12636**	—	—	—

Table 5 Comparison of average $|E_p|$ when the proposed algorithms are applied to Data1

$\|V\|$	$\|E\|$	(PDD)	(PDR)	(PDMC)
2000	6000	2102.00	2102.00	2004.60
2000	10000	2225.60	2225.60	2006.20
2000	20000	2207.20	2418.40	2006.60
2000	60000	2006.00	2955.20	2006.00
2000	100000	2006.00	3170.00	2006.00
2000	200000	2004.80	3605.60	2004.80
5000	15000	5152.40	5165.60	4926.60
5000	25000	5299.40	5299.40	5033.60
5000	50000	5467.00	5523.20	5292.40
5000	150000	5152.40	5165.60	4926.60
5000	250000	5006.60	6850.40	5346.60
5000	500000	5006.80	7459.20	7071.40
10000	30000	10102.40	10102.40	9698.20
10000	50000	10329.40	10336.60	9784.20
10000	100000	10630.00	10667.40	10200.00
10000	300000	11232.00	11572.20	10819.00
10000	500000	10241.67	12053.40	12544.50

Table 6 Comparison of average CPU time (s) when the known algorithms are applied to Data1

$\|V\|$	$\|E\|$	(DIV)	(MWW2)	(DIV2)	(PWB2)
2000	6000	44.6266	31365.1	29600.7	43.3312
2000	10000	88.3891	19132.2	26648.8	81.6547
2000	20000	217.564	—	27164	197.867
2000	60000	242.495	—	21245.4	941.112
2000	100000	301.914	—	18934	1995.02
2000	200000	320.144	—	11987	6047.73
5000	15000	275.639	—	—	242.598
5000	25000	1333.11	—	—	567.895
5000	50000	696.833	—	—	1467
5000	150000	1015.36	—	109750	7272.91
5000	250000	741.52	—	72579	—
5000	500000	940.547	—	51927.9	—
10000	30000	8031.6	—	—	1116.07
10000	50000	3604.4	—	—	2746.01
10000	100000	—	—	—	7576.21
10000	300000	1201.94	—	—	—
10000	500000	1212.19	—	—	—

by *PLAN-DIVIDE* in average, and its CPU time is 6.26% of that by *PLAN-DIVIDE* in average.

(vi) Concerning experimental results on about Data2, Data3 and Data4, *PLAN-DIVIDE* gives the best solutions for large graphs and **PDR** runs fastest of all algorithms in this experiment. **PDR** is more quickly than other algorithms. $|E_p|$ of **PDR** is 95.10% of that by *PLAN-DIVIDE* in average and the CPU time of **PDR** is 7.94% of that by *PLAN-DIVIDE* in average.

(v) It is concluded from our experimental results that **PDR** is the most useful in extracting a spanning planar subgraphs of large graphs that may appear in practical situation.

5 Concluding remarks

In this paper, we have proposed three efficient parallel

Table 7 Comparison of average CPU time (s) when the proposed algorithms are applied to Data1

$\|V\|$	$\|E\|$	(PDD)	(PDR)	(PDMC)
2000	6000	49.78	49.74	**0.34**
2000	10000	53.55	53.43	**0.45**
2000	20000	34.94	36.94	**0.75**
2000	60000	**2.02**	40.85	2.04
2000	100000	3.52	47.29	**3.50**
2000	200000	**7.90**	41.67	7.91
5000	15000	95.69	112.11	**26.70**
5000	25000	94.10	93.74	**31.02**
5000	50000	**54.70**	73.47	175.99
5000	150000	95.61	112.29	**26.76**
5000	250000	**9.09**	93.42	318.43
5000	500000	**20.20**	90.21	2598.82
10000	30000	**74.87**	75.15	280.80
10000	50000	**74.17**	91.56	745.78
10000	100000	**92.36**	115.39	1585.95
10000	300000	**102.07**	130.08	3218.06
10000	500000	**52.84**	126.97	10271.85

Table 8 Comparison of average $|E_p|$ for Data2

$\|P\|$	(DIV)	(PDD)	(PDR)	(PDMC)	(PWB2)
2000	6406.60	6366.00	6285.60	6241.40	**6511.00**
5000	**16860.00**	15586.60	15371.20	13664.40	16176.40
10000	32048.40	31085.40	30264.20	25905.80	**32238.20**

Table 9 Comparison of average $|E_p|$ for Data3

$\|P\|$	(DIV)	(PDD)	(PDR)	(PDMC)	(PWB2)
2000	9943.60	9944.00	9918.80	9517.60	**10061.50**
5000	24867.40	24643.00	24367.20	22059.80	**25024.60**
10000	49836.00	49142.00	48040.60	43284.40	**49901.40**

Table 10 Comparison of average $|E_p|$ for Data4

$\|P\|$	(DIV)	(PDD)	(PDR)	(PDMC)	(PWB2)
2000	13474.40	13481.80	13514.00	12895.00	**13600.60**
5000	**39279.40**	33514.20	33235.20	30424.00	33831.80
10000	**67300.00**	66893.40	66293.80	—	—

Table 11 Comparison of average CPU time (s) for Data2

$\|P\|$	(DIV)	(PDD)	(PDR)	(PDMC)	(PWB2)
2000	**9.51**	50.96	29.29	93.26	152.45
5000	116.25	**57.15**	101.83	744.82	959.19
10000	869.08	73.14	**54.76**	8886.42	3889.27

Table 12 Comparison of average CPU time (s) for Data3

$\|P\|$	(DIV)	(PDD)	(PDR)	(PDMC)	(PWB2)
2000	**26.05**	51.45	45.62	212.39	478.94
5000	327.97	52.33	**31.52**	3341.43	3160.57
10000	2505.98	102.90	**99.28**	27560.57	13434.93

heuristic planarization algorithms **PDD** , **PDR** and **PDMC** under forbiddance of turning over. We have evaluated performance of these three proposed algorithms and four known sequential algorithms experimentally. It is concluded that **PDR** can efficiently extract a spanning planar subgraph under forbiddance of turning over, showing usefulness for given graphs that are too large to be handled without reduction of

Table 13 Comparison of average CPU time (s) for Data4

$\|P\|$	(DIV)	(PDD)	(PDR)	(PDMC)	(PWB2)
2000	**38.15**	57.87	51.88	541.24	1006.99
5000	387.93	**61.08**	93.14	9404.67	6798.23
10000	4107.49	**139.71**	159.10	—	—

their sizes.

Incorporating proposed algorithms in a PCB layout design tool MULTI-PRIDE [18] is left for future research.

References

[1] M. R. Garey and D. S. Johnson: "Computers and Intractability: A Guide to NP-Completeness", W.H. Freeman and Co., New York (1979).

[2] J. Cai, X. Han and R. E. Tarjan: "An $O(m \log n)$-time algorithm for the maximal planar subgraph problem", SIAM J. on Computing, **22**, 6, pp. 1142–1162 (1993).

[3] G. Călinescu, C. G. Fernandes, U. Finkler and H. Karloff: "A better approximation algorithm for finding planar subgraphs", Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, New York/Philadelphia, ACM/SIAM, pp. 16–25 (1996).

[4] T. Chiba, I. Nishioka and I. Shirakawa: "An algorythm of maximal planarization of graphs", Proc. IEEE Symp. on Circuits & Sys., pp. 649–652 (1979).

[5] H. Djidjev: "A linear algorithm for the maximal planar subgraph problem", Algorithms and Data Structures, 4th International Workshop (Eds. by S. G. Akl, F. K. H. A. Dehne, J.-R. Sack and N. Santoro), Vol. 955 of Lecture Notes in Computer Science, Kingston, Ontario, Canada, Springer, pp. 369–380 (1995).

[6] O. Goldschmidt and A. Takvorian: "An efficient graph planarization two-phase heuristic", Networks: An International Journal, **24**, pp. 69–73 (1994).

[7] K. Iwamoto, T. Watanabe, T. Araki and K. Onaga: "Finding jumpers in printed wiring board design for analog circuits", Proc. of the 1991 IEEE International Symposium on Circuits and Systems, pp. 2854–2857 (1991).

[8] Y. Mizuguchi and T. Watanabe: "Application of the path-addition planarity testing algorithm to layout design of printed wiring boards", Technical Report COMP94-87, IEICE of Japan (1995 (in Japanese)).

[9] K. Mizuno, T. Kobayashi and T. Watanabe: "Extracting nonplanar connections in a terminal-vertex graph", Proc. of the 1999 IEEE International Symposium on Circuits and Systems, pp. VI–121–VI124 (1999).

[10] T. Ozawa and H. Takahashi: "A graph-planarization algorithm and its application to random graphs", Proc. of the 17th Symposium of Research Institute of Electrical Communication on Graph Theory and Algorithms (Eds. by N. Saito and T. Nishizeki), Vol. 108 of LNCS, Sendai, Japan, Springer, pp. 95–107 (1980).

[11] T. Yamaoki, S. Taoka and T. Watanabe: "Extracting a planar spanning subgraph of a terminal-vertex graph by solving the independent set problem", Proc. of the 2001 IEEE International Symposium on Circuits and Systems, pp. V–153–V–156 (2001).

[12] S. Anegayama, D. Takafuji, S. Taoka and T. Watanabe: "Hierarchical extraction of a spanning planar subgraph under fixed embedding of specified subgraphs", IPSJ SIG Notes AL80–1, IPSJ (2001). (Japanese).

[13] A. Matsumoto, K. Yamaguchi, T. Kashiwabara, S. Masuda and M. Taki: "A planarization algorithm in the layout design of single layer printed circuit board", Technical Report COMP91-21, IEICE of Japan (1991 (in Japanese)).

[14] K. Kotani, S. Masuda and T. Kashiwabara: "An alogrithm for embedding a biconnected planar graph to to maximize the total sum of vertex-and edge-weights on the exterior window", Trans. IEICE, **J74-A**, 7, pp. 1041–1052 (1991 (in Japanese)).

[15] T. Ozawa: "Efficient algorithms for planar embedding of graphs with constraints in placing specified vertices on face boundaries", Proc. of the 2002 IEEE International Symposium on Circuits and Systems, pp. V–749–V–752 (2002).

[16] T. Ozawa and H. Takahashi: "A graph planarization algorithm and its application to random graphs", In Graph Theory and Algorithms, Lecture Notes in Computer Science, **108**, pp. 95–107 (1981).

[17] D. Takafuji, S. Anegayama, T. Kinoshita and T. Watanabe: "Heuristic algorithms for extracting a planar graph with subgraphs forbidding their turning over", IPSJ SIG Technical Report 2003-AL-91 (1), IPSJ (2003).

[18] T. Watanabe: "MULTI-PRIDE: A system for supporting multi-layered printed wiring board design", Proc. of ASP-DAC '97, pp. 221–226 (1997).