Enhanced Heuristic Algorithms K-LAG-V and K-LAG-S for the Constrained Via Minimization Problem

Daisuke Takafulji, Toshimasa Watanabe and Yuji Suga
1 Graduate School of Engineering, Hiroshima University
1-4-1, Kagamiyama, Higashi-Hiroshima, 739-8527 Japan
Email: {takafulji, watanabe, suga}@infonets.hiroshima-u.ac.jp

Abstract: CVM requires finding any layer assignment of wire-segments, whose topology has already been given, so that the total number of vias may be minimized. A given topology of wire-segments is called an initial wiring layout. Let kCVM denote CVM in which k layers are available for routing. In this paper, only rectilinear routing is considered. The subject of the paper is to propose heuristic algorithms K-LAG-V and K-LAG-S that are enhanced versions of K-LAG. Based on experimental results, it is shown that they are promising ones for solving kCVM with k ∈ {4, 12}.

1. Introduction

A via is a through-hole used in connecting wire-segments placed on different layers in layout design of VLSI chips or printed circuit boards. Vias may cause higher production cost, larger routing area, or deterioration of reliability because of increased resistance and delay, and, therefore, designing layouts with the smallest possible number of vias is required. The problem, which decides topology and the layer assignment of wire-segments so that the number of vias in the layout are minimized, is called “the via minimization problem”.

There are two types of via minimization problems: the unconstrained via minimization problem (UVM) and the constrained via minimization problem (CVM). UVM asks for deciding both topology and the layer assignment of wire-segments at the same time, while CVM requires finding any layer assignment of wire-segments, whose topology has already been given. Topology of wire-segments is called a wiring layout, and initially given topology of wire-segments is called an initial wiring layout. Let kCVM denote CVM in which k layers are available for routing. In this paper, only rectilinear routing (that is, only horizontal or vertical wiring) is considered.

Some of known results on CVM are summarized. For 2CVM, polynomial time algorithms [3, 5, 6], or a heuristic one [4] has already been proposed. It is known in [2] that kCVM with k ≥ 3 is NP-hard even if k = 3, and a heuristic algorithm is given in [1]. K-LAG [1] is known to have very high capability for solving general kCVM.

In this paper, we propose heuristic algorithms K-LAG-V and K-LAG-S for kCVM, which are enhanced versions of K-LAG, and evaluates their performance through computing experiments.

2. Definitions

The paper considers designing one printed circuit board consisting of several layers, each can be used for routing, and any module is to be placed only on the top or bottom layer. In kCVM, we assign each of k layers an identification (ID) number 1, . . . , k from the top to the bottom.

Given a set of terminals, a net is a set of terminals such that they are acyclically connected by wires. A via-candidate is a location (to be represented as a point) such that either a via is assigned to it in a given initial wiring layout or a junction of a wire for some net, where a junction is a grid point at which the wiring bends. Let us call terminals and via-candidates as endpoints. A wire-segment is a portion of a wire between two endpoints such that no intermediate endpoints are included. There are two types of via minimization problems: the Constrained Via Minimization Problem (CVM) asks that the total number of vias may be minimized. A given net is to be represented as a point) such that either a via is assigned to it in a given initial wiring layout or a junction of a wire for some net, where a junction is a grid point at which the wiring bends. Let us call terminals and via-candidates as endpoints. A wire-segment is a portion of a wire between two endpoints such that no intermediate endpoints are included.

3. A Known Heuristic Algorithm K-LAG

In this section, a heuristic algorithm K-LAG [1] for kCVM is explained.

Some additional definitions are given. An ECC graph [1] of a given wiring layout is defined as follows:

(a) V ← VC ∪ VW, where VC is the set of all via-candidates, and VW is the set of vertices each representing a wire-segment.
(b) E consists of edges defined by (i) or (ii).
   (i) A continuation edge: one endvertex is representing a wire-segment and the other is representing a via-candidate which is one of endpoints of this wire-segment.
   (ii) A conflict edge: two endvertices are included in different layers, and they are placed at the same location in the layout projected onto one layer.

Let ET or EF denote the set of continuation edges or of conflict edges, respectively, where E = ET ∪ EF.

K-LAG first constructs an ECC graph of a given initial wiring layout, and then repeats deciding layer assignment of wire-segments, represented as vertices of this ECC graph, so that the subgraph induced by these vertices may be a tree. The formal description is given as follows.

Algorithm K-LAG
Input: An initial wiring layout σ and a terminating condition σ


1529
t (a real number);
Output: A wiring layout $\sigma^*$ for $\sigma$;
Step 1. Find all via-candidates in $\sigma$.
Step 2. $p \leftarrow 1$. Construct an ECC graph $G = (V, E)$ of $\sigma$.
Step 3. Set every vertex as “unvisited”. If $p = 1$ then $\sigma^* \leftarrow \sigma$ and $\sigma' \leftarrow \sigma$.
Step 4. Execute (a) through (d) until all vertices of $G$ are changed to “visited”.
(a) Choose any “unvisited” vertex $r$. By applying Function AS to $G$ and $r$, find a maximal tree component $V' \subseteq V$ rooted at $r$. Set every vertex $v \in V'$ as “visited”.
(b) For each $w \in V'$, decide all layers to which $w$ can be assigned or to which $w$ cannot be assigned.
(c) Select layer assignment that minimizes the total number of vias to be assigned to via-candidates represented by vertices in $V'$.
(d) Update $\sigma^*$ by checking layer assignment for each $w \in V'$.
Step 5. Compute the via reduction rate $D_\sigma^* = \frac{n(\sigma^*)-n(\sigma)}{n(\sigma^*)} \times 100$. If $D_\sigma^* > t$ then $\sigma' \leftarrow \sigma^*$ and return to Step 3 else output $\sigma^*$ and halt.

Function AS (arbitrary selecting)
1. $V' \leftarrow \{r\}$.
2. While $Ad(V') \neq \emptyset$, execute (a) and (b).
(a) Choose any vertex $v \in Ad(V')$.
(b) If $V' \cup \{v\}$ is a tree component of $G$, then $V' \leftarrow V' \cup \{v\}$ and update $Ad(V')$, else $Ad(V') \leftarrow Ad(V') - \{v\}$.

Example 1: Given an initial wiring layout in Fig. 1, the ECC graph $G$ shown in Fig. 2 is constructed, and the layer assignment as in Fig. 3 is obtained from Fig. 1 and Fig. 2 by means of $K-LAG$.

4.1 Introducing Functions $SV$ and $SC$

The point of $K-LAG-V$ or $K-LAG-S$ is that we try to select a vertex $v \in Ad(V')$ for enlarging $V'$ in Step 4(a) so that the resulting $V'$ may satisfy the following condition.

(K-LAG-V) $V'$ contains no less vias representing via-candidates than any vertex set obtained by $K-LAG$.

(K-LAG-S) The tree induced by $V'$ contains no less continuation edges than the tree induced by any vertex set obtained by $K-LAG$.

$K-LAG-V$ or $K-LAG-S$ is an enhanced version of $K-LAG$ by using Function $SV$ or $SC$, instead of Function $AS$, respectively. Functions $SV$ and $SC$ are described as follows. Their differences from $AS$ exist in the following Step 2(a), and the rest is the same. Hence only Step 2(a) is shown.

Function $SV$ (priority to selecting via-candidates)
$Step 2(a)$ If $Ad(V') \cap V_C \neq \emptyset$, then select $v \in Ad(V') \cap V_C$ else select $v \in Ad(V')$.

Function $SC$ (priority to selecting continuation edges)
$Step 2(a)$ If $K(V', Ad(V')) \subseteq E$ contains at least one continuation edge, then (i) select any continuation edge and (ii) select the vertex $v \in Ad(V')$, which is an end-vertex of the selected edge, else select $v \in Ad(V')$.

4.2 Expected effects of $SV$ and $SC$

Let $x_1, x_2, y$ be any three vertices such that $x_1, x_2 \in V_W$, $y \in V_C$, $(y, x_1) \in E_T$ and $(y, x_2) \in E_T$. If both $x_1$ and $x_2$ are assigned to the same layer (that is, $L(x_1) = L(x_2)$), then no vias are assigned to $y$. Even if $L(x_1) \neq L(x_2)$, it is desirable to make $|L(x_1) - L(x_2)|$ as small as possible, since...
this may decrease $\delta(y)$. Hence simultaneously changing layer assignment of $x_1$ and $x_2$ has higher possibility of avoiding increase in $\delta(y)$. Layer assignment of each vertex $v \in V_C \cup V_V$ is changed only if it is included in $V'$. If one of $x_1$ and $x_2 \in V_V$ is not included in $V'$, then $|L(x_1) - L(x_2)|$ may be increased, which may result in increase in $\delta(y)$, and this should be avoided.

Therefore we try to find a tree component $V'$ containing as many elements in $V_C \cup E_T$ as possible, as in Functions SV and SC.

Example 2: Fig. 4, Fig. 5 or Fig. 6, respectively, shows a pair of a tree component $V'$ rooted at $a$ and the subgraph induced by $V'$, which are obtained by Function AS, SV or SC for the ECC graph of Fig. 2. The tree of Fig. 5 contains more vertices representing via-candidates than that of Fig. 4, and the tree of Fig. 6 contains more continuation edges than that of Fig. 4.

It is shown through experiment results that Functions SV and SC improve capability of algorithms. Comments on another Function SVC, a hybrid version of SV and SC, is given in 5.3.

5. Experiment Results

The existing algorithm K-LAG and the proposed algorithms K-LAG-V and K-LAG-S have been implemented on a personal computer (CPU: Opteron 285, OS: FreeBSD6.2-RELEASE) with the C programming code. By using random numbers, we constructed input problems to CVM such that sets of nets and feasible HVH $k$-layer wiring layouts ($k = 4, 12$) with 500 to 1,500 nets. In order to compare capability of these algorithms, we pay attention to the via reduction rate $D_v(\%)$ defined by:

$$D_v = \frac{n(\sigma) - n(\sigma^*)}{n(\sigma)} \times 100$$

such that the greater percentage, the better, where $\sigma$ ($\sigma^*$, respectively) is a given initial layout (the layout obtained by each algorithm). Also compared are CPU time in second, sizes $|V'|$ and the numbers of via-candidates and continuation edges.

5.1 Input problems and experiment results

(Input problems) The total number of input problems is 300. The number of nets, $\sharp\text{NET}$, is 500 $\leq \sharp\text{NET} \leq 1,500$, where each net is a 2-terminal net.

(Experimental results) Tables 1 through 3 show a part of experiment results, where each algorithm has a terminating condition $t = 0$.

Via reduction rates $D_v(\%)$ and CPU time (second) over 300 input problems are summarized in Table 1, where the minima, the averages and the maxima are denoted as $\text{min}/\text{ave}/\text{max}$ in the table.

Let $E(G[V'])$ denote the edge set of the subgraph induced by $V'$ of $G$ for any $V' \subseteq V$. We compare the averages of the three ratios $|V'|/|V |$, $|V' \cap V_C|/|V'|$ and $|E(G[V'])|/|E_T|$ over all tree components $V'$ obtained in Step 4(a) of the algorithms K-LAG, K-LAG-V and K-LAG-S in Table 2. Let $\sharp\text{LO}$ be the number of iteration of Steps 3 through 5 in K-LAG, K-LAG-V or K-LAG-S, and $\sharp\text{TR}$ be the total number of tree components found during execution of each algorithm. Let $\text{Ave}(\cdot)$ denote their averages. Table 3 shows the ratio $\text{Ave}(\sharp\text{TR})/\text{Ave}(\sharp\text{LO})$, which shows the average number of tree components found in each execution of Steps 3 through 5, in the period from the beginning until all vertices are set “visited” in Step 4.

5.2 Observations

The left half of Table 1 shows that the average via reduction rates of K-LAG-V and K-LAG-S are 0.48% and 2.32% higher than that of K-LAG, respectively. On the other hand, CPU time of K-LAG-V and K-LAG-S is 1.78 times and 1.99 times that of K-LAG, respectively, as shown in the right half of Table 1.

Table 2 shows averages of ratios $|V'|/|V |$, $|V' \cap V_C|/|V'|$ and $|E(G[V'])|/|E_T|$ obtained by K-LAG-V, K-LAG and K-LAG-S, respectively. The results clearly show the effect of adopting Functions SV and SC. Comparing the results of Table 1 suggests that finding a tree component $V'$ for which $|E(G[V'])|/|E_T|$ is as large as possible gives us highest via reduction rates.

Table 3 shows the ratios $\text{Ave}(\sharp\text{TR})/\text{Ave}(\sharp\text{LO})$ obtained by K-LAG, K-LAG-V and K-LAG-S, where K-LAG-S has the largest ratios. That is, K-LAG-S finds the largest number of
Table 1. Comparison of $D_v$ (%) (left half) and CPU time (s) (right half), where minima, averages and maxima are shown as min/average/max, where \( z \) is the number of available layers and \( \# \) is the total number of nets.

<table>
<thead>
<tr>
<th>( z )</th>
<th>( # )</th>
<th>Via reduction rates $D_v$ (%)</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
<td>1500</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>K-LAG</td>
<td>21.7/24.62/25.90</td>
<td>20.88/22.45/24.01</td>
</tr>
<tr>
<td></td>
<td>K-LAG-V</td>
<td>21.49/24.01/26.59</td>
<td>21.25/22.76/24.22</td>
</tr>
<tr>
<td></td>
<td>K-LAG-S</td>
<td>22.64/24.82/27.54</td>
<td>21.96/23.58/25.07</td>
</tr>
<tr>
<td>12</td>
<td>K-LAG</td>
<td>31.7/33.96/35.98</td>
<td>28.09/29.68/31.12</td>
</tr>
<tr>
<td></td>
<td>K-LAG-V</td>
<td>32.41/34.19/36.67</td>
<td>28.49/29.85/31.19</td>
</tr>
<tr>
<td></td>
<td>K-LAG-S</td>
<td>35.07/36.81/39.04</td>
<td>29.85/31.86/33.52</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the average of $|V'_C|/|V_C|$, $|V'_C|/|V_C|$ and $|E(G)|/|E_T|$.

<table>
<thead>
<tr>
<th>( z )</th>
<th>( # )</th>
<th>Via reduction rates $D_v$ (%)</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
<td>1500</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>K-LAG</td>
<td>47.31</td>
<td>46.17</td>
</tr>
<tr>
<td></td>
<td>K-LAG-V</td>
<td>46.24</td>
<td>45.27</td>
</tr>
<tr>
<td></td>
<td>K-LAG-S</td>
<td>44.85</td>
<td>44.96</td>
</tr>
<tr>
<td>12</td>
<td>K-LAG</td>
<td>44.85</td>
<td>44.96</td>
</tr>
</tbody>
</table>

Table 3. Comparison of the ratios $\text{Ave}(2TR)/\text{Ave}(2LO)$.

<table>
<thead>
<tr>
<th>( z )</th>
<th>( # )</th>
<th>Via reduction rates $D_v$ (%)</th>
<th>CPU time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
<td>1500</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>K-LAG</td>
<td>74.51</td>
<td>78.06</td>
</tr>
<tr>
<td></td>
<td>K-LAG-V</td>
<td>104.98</td>
<td>108.06</td>
</tr>
<tr>
<td></td>
<td>K-LAG-S</td>
<td>64.43</td>
<td>68.13</td>
</tr>
<tr>
<td>12</td>
<td>K-LAG</td>
<td>87.95</td>
<td>111.67</td>
</tr>
</tbody>
</table>

6. Concluding Remarks

Heuristic algorithms K-LAG-V and K-LAG-S for $k$CVM, which are enhanced versions of K-LAG [1], are proposed. Their performance is evaluated through computing experiments, and it is concluded that K-LAG-S is more useful in practical situations than the algorithms K-LAG and K-LAG-V.

Proposing another new algorithm and comparing capability of algorithms for larger input problems are left for future research.

Acknowledgments

The research is partly supported by the Grant-in-Aid for Scientific Research (C) (No. 20500015) of the Ministry of Education, Culture, Sports, Science and Technology.

References


