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

2008－SLDM－135（8）
2008／5／9

障害物を考慮した高速配線長推定手法

リシューティン† ヤン タン†† 高島 康裕† 村田 洋†

† 北九州市立大学国際環境工学部情報メディア工学科
〒 808–0135 北九州市若松区ひびきの 1 － 1
†† イリノイ大学 アーバナ・シャンペイン校
E-mail: †zoe.lee@is.env.kitakyu-u.ac.jp, ††tanyan2@uiuc.edu, †††{takasima,hmurata}@env.kitakyu-u.ac.jp

あらまし IP 再利用等においては，利用する IP の特性を考慮して設計を行なう必要がある．特にフロアプランにおいては，近傍の配線の状況の影響を受けやすい IP は障害物として考慮する必要がある．しかし，従来の HPWL 方式では，障害物の考慮を評価中に組み込むのは不可能であった．本稿では，ヤンらが [5] において提案した手法を基に，障害物を考慮した最小配線長評価アルゴリズムの改良を検討し，より実用的な近似手法を提案する．実験により，提案手法が HPWL 評価手法と比較して，数倍程度の時間で実現できることを確認した．

キーワード visibility graph, lookup-table, 有向非循環グラフ，最短経路

Fast Wire Length Estimation in Obstructive Block Placement

Shuting LI†, Tan YAN††, Yasuhiro TAKASHIMA†, and Hiroshi MURATA†

† Faculty of Environmental Engineering
Department of Information and Media Sciences
The University of Kitakyushu
Hibikino 1–1, Wakamatsu, Kitakyushu,808–0135 Japan
†† Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, Illinois 61801, USA
E-mail: †zoe.lee@is.env.kitakyu-u.ac.jp, ††tanyan2@uiuc.edu, †††{takasima,hmurata}@env.kitakyu-u.ac.jp

Abstract IP-reuse can enhance the design productivity only if the design methodology treats the IPs in a proper way. Especially in the floor-planning phase, sensitive IPs should be treated as routing obstacles, which is impossible when the conventional HPWL-based method is used for routing estimation. This paper proposes an obstacle-aware minimum wiring length (MWL) estimation algorithm, based on the theoretical result in [5], through algorithmic improvements and practical approximation. The experimental results suggests that MWL-based estimation is now possible with only few times larger computational cost comparing to the HPWL-based estimation.

Key words visibility graph, lookup-table, directed-acyclic graph, shortest path

1. Introduction

In modern LSI design methodologies, the gap between productivity and designability became a hard problem, and IP-reuse is expected to be the key to solve this crisis. To let IP-reuse being successful, the design methodology should pay detailed care on the reusing IPs. Especially in the first and most important floor-planning phase, it is often the case that the sensitive IPs should be treated as routing obstacles. The conventional *half-perimeter-wire-length* (HPWL)-based estimation, is far insufficient to handle such obstacle-aware routings(see Fig.1). Recently in [5], based on the computational geometry research by Mitchell [6], it is shown that obstacle avoiding minimum wire length (MWL) of two-pin nets can be estimated in $O(n)$ time per one net, as well as in $O(1)$ time per one net with $O(n^2)$ time preprocess for all nets, where n is the number of obstacles. However, they give no experimental result, and the computational cost of their methods might be too heavy for practical use.

In this paper, we first improve their preprocess algorithm to cut out repeated calculation observed in our experiments on certain edges in constructing the look-up table, but unfortunately experiments show that even our improved version of the algorithm might be still too slow. A breakthrough is obtained by the observation that routing detours around obstacles tend to occur near the terminals. Using this observation, we finally propose approximation algorithm with no preprocess, that considers routing detours only around terminals. The experimental results show that the calculation time of the proposed algorithm is only few times longer than HPWL-based method, while the degradation on the accuracy is tolerable.

The rest of this paper is organized as follows: section 2. describes the problem we deal with. section 3. presents our estimation methods. section 4. gives the experimental results. Finally, section 5. concludes the paper.

2. Problem Setup

We define the wire length estimation with obstructive blocks problem as follows:

Input :

(1) The placement. (Including the locations of blocks and pins, and the information of nets.)

(2) The ABLR relations [1] of the blocks.

Constraints :

(1) The blocks are non-overlapping rectangles.

(2) The pins are located at the peripheries of the blocks.

(3) The routing adopts the Manhattan geometry.

(4) All the blocks are obstacles for routing.

Output :

The length of the shortest routing of a 2-pin net that avoids all the blocks.

The ABLR relation is universal for relation-based representations, e.g., Sequence-Pair [2]. It indicates whether a block A is located *above*, *below*, *left-to*, or *right-to* another block B. If A is above B, then A's lower boundary is above B's upper boundary. If other representations, e.g., B*-tree [3], are used, we can translate the placement into a sequence-pair in $O(M \log M)$ time [4] and then obtain the ABLR relation. The fourth constraint is for the simplicity of discussion. It will be shown later that the proposed method can be easily extended to handle the problem in which only selected blocks are obstacles.

In this paper, we use (s, t) to denote a 2-pin net with s as the source pin and t as the sink pin. Without loss of generality, we assume the net satisfies:

[Assumption 1] s is on block S and t is on block T, $S \neq T$. S is left-to T. $y_s \leq y_t$ (s is located lower than t).

Fig.1 gives an example of such a net. This paper discusses only the situation that satisfies Assumption 1. Other situations such as S is above T or $y_s > y_t$ can be discussed by vertically/horizontally flipping the placement or rotating the placement by 90°.

[Definition 1] The length of the shortest routing that avoids all the blocks is called the *minimum wire length* (MWL) of the net. A routing with this length is called an *MWL routing*.

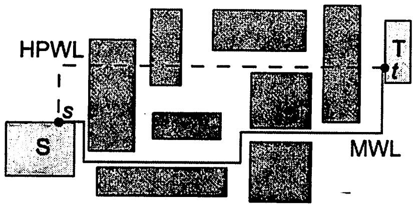


Figure 1 The HPWL may underestimate the Minimal Wire Length (MWL) of an actual routing.

3. Algorithms

In 3. 1 and 3. 2, we briefly review the algorithms in [5]. In 3. 3, we propose the *Speedup Algorithm of LUT*. Finally, an approximation method that only considers the detours around terminals is proposed in 3. 4.

3. 1 *Visibility Graph* (VG) Algorithm

How to obtain the MWL routing by finding the shortest path on a DAG is introduce in [5]. The key is to construct the DAG. They also defined the VG (VG can be classified to *Horizontal Visibility Graph* (HVG) and *Vertical Visibility Graph* (VVG)). It is necessary to construct both of them when calculate the MWL. But we only discuss the HVG for conciseness. VVG can be easily constructed by connecting edges vertically. HVG can be built according to [5], HVG is a *directed-acyclic graph* (DAG) from left to right (see Fig.2).

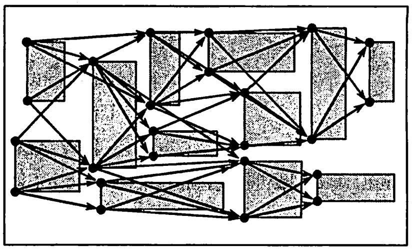


Figure 2 The HVG of a placement.

The MWL of any two vertices on the VG can be obtained by using *Shortest Path Algorithm* according to [5]. It has been proved that if there exists a shortest path on VG, the MWL is the length of the path. If there is not any existing path, the MWL is equal to HPWL. Now, we can summarize the VG Algorithm (see Algorithm.3. 1).

The total time complexity of Algorithm.3. 1 is $O(M \log M + NM)$ according to [5], if using a *Plane Sweep Method* to construct VG. This is much better than the $O(NM^2)$ complexity if we apply *Dijkstra's Algorithm* on the *Channel Intersection Graph.*

(注1) : means ABLR-relation (Above-Below relation or Left-Right relation), which mainly base on the relationship of the blocks that s and t are on.

(注2) : the edges go from pin s, and the edges go to pin t

Algorithm.3. 1 *Visibility Graph* (VG)

```
1 : Construct the VG $G_P$ of the placement $P$
2 : $TotalWL = 0$
3 : for each 2-pin net $(s, t)$ do
4 :    Judge the relationship(注1) between $s$ and $t$.
5 :    Add pin edges(注2).
6 :    If $s$ and $t$ is Left-Right (LF) Relation then
7 :       Search the shortest path from $s$ to $t$ on HVG.
8 :       if there exists no path then
9 :          $MWL = HPWL$
10:       end if
11:    else
12:       Search the shortest path from $s$ to $t$ on VVG.
13:       if there exists no path then
14:          $MWL = HPWL$
15:       end if
16:    end if
17:    $TotalWL = TotalWL + MWL$
18:    Remove pin edges.
19: end for
```

3.2 *Lookup Table* (LUT) Algorithm

We can record note-to-node distances in a LUT and use it during estimation phase. When estimating a net (s, t) (see Fig.3), we just select the shortest path among the four choices $(s \rightarrow a_1 \rightarrow b_1 \rightarrow t)$, $(s \rightarrow a_1 \rightarrow b_2 \rightarrow t)$, $(s \rightarrow a_2 \rightarrow b_1 \rightarrow t)$, and $(s \rightarrow a_2 \rightarrow b_2 \rightarrow t)$. Since all the distances and the edges are pre-calculated, this takes only a constant time.

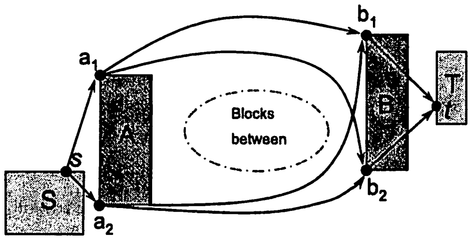


Figure 3 The four paths in LUT Algorithm.

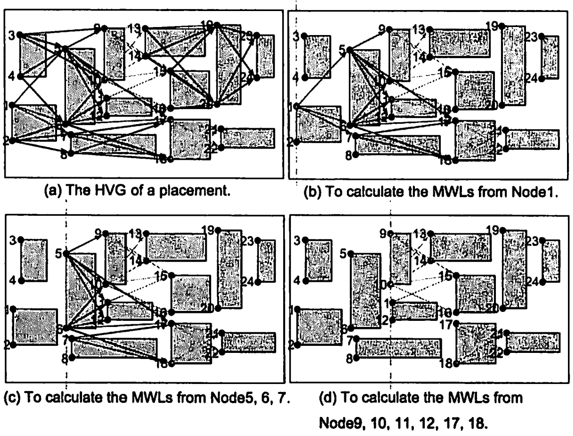
Algorithm.3. 2 *Lookup Table* (LUT)

```
1 : Construct the LUT $G_P$ of the placement $P$
2 : $TotalWL = 0$
3 : for each 2-pin net $(s, t)$ do
4 :    Add pin edges.
5 :    Select the shortest path among the four choices
6 :    $TotalWL = TotalWL + MWL$
7 :    Remove pin edges.
8 : end for
```

We present LUT based MWL estimation algorithm (see Algorithm.3. 2). The total time complexity of Algorithm.3. 2 is $O\left(N + M^2\right)$ according to [5].

3.3 *Speedup Algorithm of LUT*

Before we discuss the *Speedup Algorithm of LUT*, let us review LUT Algorithm. As an example of how to construct the LUT, we sort the nodes from left to right, and we have already constructed the HVG(as shown in Fig.4 (a)). Then we calculate the MWL from nodes to nodes by *Shortest Path Algorithm* on DAG, using the order we sorted. First, the MWLs from $Node1$ are calculated (as shown in Fig.4 (b)), notice that the MWL between $Node1$ and $Node3$ can be calculated by HPWL because there is no horizontal edge. When we calculate the MWLs from $Node5, 6, 7$ (as shown in Fig.4(c)), they have already been calculated once. Furthermore, when we calculate the MWLs from $Node9, 10, 11, 12, 17, 18$ (as shown in Fig.4 (d)), they are calculated three times. To repeat calculating the MWLs from 1 node is a waste of time, obviously.



(a) The HVG of a placement.
(b) To calculate the MWLs from Node1.
(c) To calculate the MWLs from Node5, 6, 7.
(d) To calculate the MWLs from Node9, 10, 11, 12, 17, 18.

Figure 4 An example of how to construct the LUT.

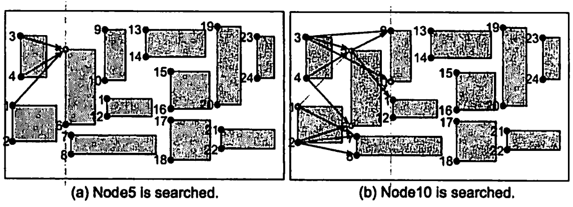


Figure 5 An example of applying the *Speedup Algorithm of LUT*.

In the *Speedup Algorithm of LUT*, we still search the shortest path from left to right, an example is shown in Fig.5. First, give all MWLs an infinite value. But this time, we do not check whether there is any edge from the current node. Instead, we check whether there is any edge to the current node. If there is an edge to the current node, update the MWL to this node. As shown in Fig.5 (a), when $Node5$ is searched, there are 3 edges to $Node5$. Therefore, the MWLs of $(Node1, Node5)$, $(Node3, Node5)$, and $(Node4, Node5)$ are updated by HPWL of them, respectively. When $Node10$ is searched (see Fig.5 (b)), the MWLs of $(Node5, Node10)$ and $(Node6, Node10)$ are updated since there exists edges. If we want to calculate the MWL of ($Node1, Node10$), since the MWLs of ($Node1, Node5$) and ($Node1, Node6$) are already known, we can compare the value ($Node1, Node5$) + ($Node5, Node10$) and the value ($Node1, Node6$) + ($Node6, Node10$), then choose the shorter one for the MWL of ($Node1, Node10$). Now, we summarize the *Speedup Algorithm of LUT* on the stage of calculating MWL between nodes (see Algorithm.3. 3).

Algorithm.3. 3 Speedup Algorithm of LUT

```
1 : for each MWL do
2 :   $MWL = \infty$
3 : end for
4 : for each node $t$ sorted do
5 :   for each $t$'s parent node(注3) $p$ do
6 :     $MWL(p,t) = HPWL(p,t)$
7 :   end for
8 :   for each $s$ in front of $t$ (in the sorted order) do
9 :     for each $t$'s parent node $p$ do
10:       if ($MWL(s,p) \neq \infty$) &&
            ($MWL(s,t) > MWL(s,p) + MWL(p,t)$)
          then $MWL(s,t) = MWL(s,p) + MWL(p,t)$
11:       end if
12:     end for
13:   end for
14: end for
15: for each $MWL(s,p) \neq \infty$ do
16:   $MWL = HPWL$
17: end for
```

As you see, we do not need to repeat calculating the MWLs from one node if we use the Algorithm.3. 3. Although there is no change of time complexity, but we can still expect *Speedup Algorithm of LUT* runs faster than LUT.

3.4 *No Preprocess* (NP) Algorithm

All the algorithms up to now target at exact obstacle avoiding wire length. However, in our experiments, we observe that they are usually very slow compared to HPWL in practice. If we aim at speed, we have to abandon the lookup table which is a preprocess every time we want to calculate the MWLs of a placement.

When a placement is optimized or nearly optimized, modules that are connected tend to be pulled together, leaving less possibilities for detouring. When two modules are very close, their node-to-node distance in the look-up table is the HPWL (see Fig.6(a), distances between node a_i and b_i are just HPWL). Therefore, we propose the fast non-preprocess algorithm which loses exactness only in a very minor degree. In NP Algorithm, we take HPWL for ($a_1 \rightarrow b_1$), ($a_1 \rightarrow b_2$), ($a_2 \rightarrow b_1$), and ($a_2 \rightarrow b_2$) to calculate $s - to - t$ length. Then take the shortest one for estimation. Notice that in Fig.6, since block S and block T is *left-right* relation, detours occur around the horizontally adjacent blocks of terminals. When the relation

(注3) : means the node connecting to node t directly.

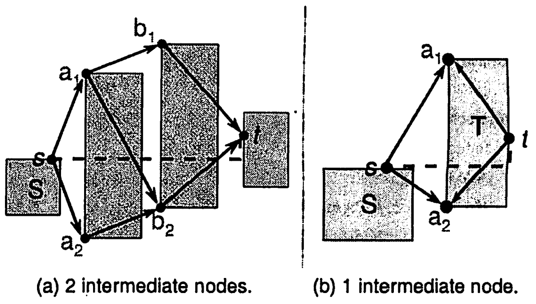


Figure 6 If all the paths from s to t have less than 2 intermediate blocks, then NP estimation is exact while HPWL estimation may still contain huge error.

of S and T is *above-below* relation, the a_1, a_2, b_1, b_2 stand for the nodes of vertically adjacent blocks. The algorithm is shown in Algorithm.3. 4.

Algorithm.3. 4 *No preprocess* (NP)

```
1 : $TotalWL = 0$
2 : for each 2-pin net $(s,t)$ do
3 :   Add pin edges.
4 :   Calculate the HPWL of ($a_1 \rightarrow b_1$), ($a_1 \rightarrow b_2$),
          ($a_2 \rightarrow b_1$) and ($a_2 \rightarrow b_2$).
5 :   Select the shortest path among the four choices
6 :   $TotalWL = TotalWL + MWL$
7 :   Remove pin edges.
8 : end for
```

It is obvious that when every path from s to t has 2 or less intermediate blocks, our NP estimation is guaranteed to be exact. Therefore, the NP estimation is always exact when the two blocks S and T are adjacent and is exact with high possibility when they are very close. On the other hand, directly using HPWL to estimate the wire length may still suffer from huge error even when the blocks are close. See Fig.6 (b) for an example, even though the two blocks are adjacent, HPWL estimation ignores T's height.

NP Algorithm avoids the time consuming "shortest path computation" and is therefore expected to be very fast.

We did an experiment to show the average errors of 5 trials during SA, using ami33 benchmark, while using exact MWL length for SA cost function. We take average of the last 50 counts(within 110% of the chip area). The average error wire length between exact wire length of all the 2-pin nets and the wire length only considering the detours around terminals is 401, while the average error between exact wire length and the length estimated by HPWL is *1630*. We can see that NP gives much less error than HPWL.

4. Experiments

We use *Simulated Annealing* (SA) to optimize the block placement for both area and wire length. Fig.7 shows a block placement on the chip, the smallest rectangle that can cover all blocks is called bounding box. While, *bounding$_w$* and *bounding$_h$* stand for the weight and height of the bounding box, respectively. *chip$_w$* and *chip$_h$* stand for the weight and height of the chip, respectively. The target of SA is to minimize the following function (1).

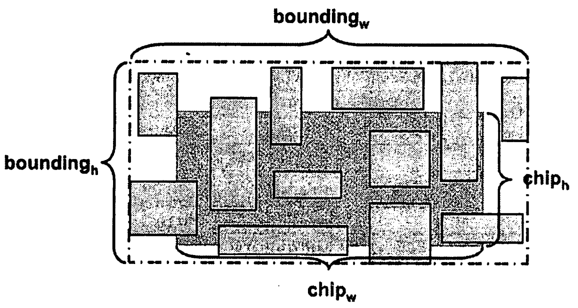


Figure 7 The chip and bounding box of a placement.

$$
\begin{aligned}
total_{cost} &= wire_{weight} * wl_{cost} + (1 - wire_{weight}) * area_{cost} \\
area_{cost} &= \frac{MAX(0, bound_w - chip_w)}{chip_w} + \frac{MAX(0, bound_h - chip_h)}{chip_h} \\
wl_{cost} &= wl / (net_{no} * (chip_w + chip_h)) \qquad (1)
\end{aligned}
$$

Where *wl* stands for the total wire length of all the nets (we take the HPWL for the multi-pin nets), *net$_{no}$* is the number of all nets. In the experiment, *wire$_{weight}$* is set to 0.5, so we estimate by half of area cost and half of wire length cost.

4.1 Experiments on different estimations

Table 1 is the experimental results of cite MCNC & GSRC applying SA. The results are the averages of 10 trials. The rows list the results of applying different algorithms in SA. The third row are the results of VG Algorithm, LUT Algorithm and *Speedup Algorithm of LUT*. Since the block placements are generated by the same seeds, the results are the same as each other (except the time). The wire length results based on the final placements of applying SA. The first column are the results of final placement calculating 2-pin net wire length using HPWL model, the second column used NP, and the third used LUT to calculate the exact wire length of 2-pin nets. The fourth column is the number of detour nets. The fifth column shows the percentage that the exact wire length and the wire length estimated by HPWL differs by. The sixth column shows the percentage that the exact wire length and the wire length estimated by NP differs by. The seventh column is the wire length of multi-pin nets using HPWL model. The last column shows the timings during SA.

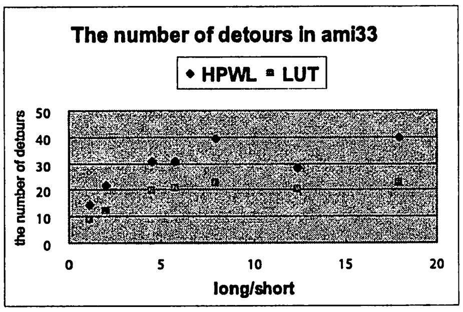
We can see using algorithms that we proposed to calculate the exact wire length of the placement during SA obtained the better placement results. Because the wire length is shorter than the result applying HPWL, the wire length of detours is the least.

Note the runtime shown in Table 1 on calculating exact wire length, in ami33 benchmark, VG spent the least time. But as the number of blocks increased, VG becomes the slowest. LUT is much better than VG. When the number of blocks becomes more and more, the *Speedup Algorithm of LUT* showed its efficiency. But it is still over 20 times longer than HPWL. The NP Algorithm consumes only several times longer than the HPWL-based method, while the degradation of the accuracy is little in ami33 and ami49 benchmarks. This verified that detours around obstacles tend to occur near the terminals.

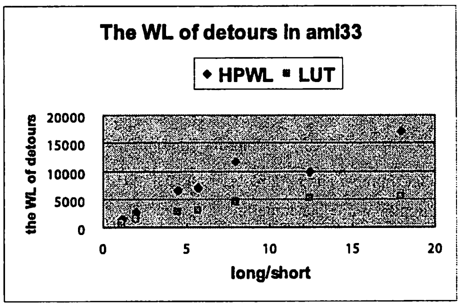
However, in n100 benchmark, NP obtained a worse exact wire length than HPWL. We noticed that the difference between the exact wire length and the wire length estimated by HPWL is not that much as ami33 and ami49 meanwhile. We found that the blocks in n100 benchmark are rectangles close to square. The average aspect ratio of all the blocks in n100 is 1.71, while the aspect ratio of ami33 and ami49 are 2.02 and 2.20, respectively. Therefore, we verify the aspect ratio of blocks affect detour a lot in the next experiment.

4.2 Experiment on how the blocks' aspect ratio affects detour nets

We take ami33 and ami49 benchmarks to do this experiment. The experiment keeps the area of block, and change the widths and heights of the blocks. The experimental results are shown in Fig.8 and Fig.9.



(a) The number of detours



(b) The length of detours

Figure 8 The number and length of detours using HPWL and LUT estimation in ami33 benchmark.

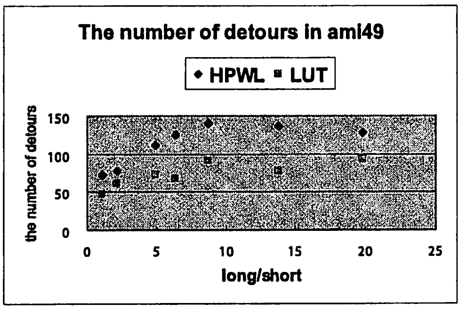
The horizontal axis is the average aspect ratio of the blocks The results of the number and the wire length of the detour nets are the average results of five trials. We observed that the larger the aspect ratio is, the more the detours are, and the longer the lengths of detours are. Using HPWL underestimate the wire length, as the aspect

Table 1 Experimental Results for Benchmarks

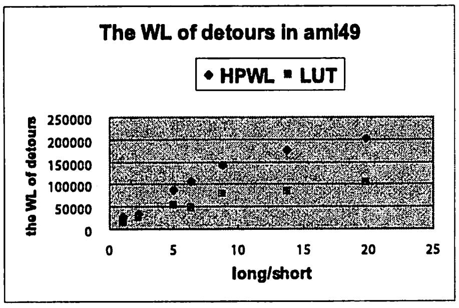
ami33	104 2-pin nets					multi	time (s)		
	HPWL	ExactWL	# detour	(ExactWL-HPWL)/Exact(%)	(ExactWL-NP)/Exact(%)	HPWL	VG	LUT	Speedup
HPWL	52234	54707	20	4.52	-	22829	15		
NP	50252	51629	14	2.67	0.81	22697	64		
VG& LUT& S	51623	52864	12	2.35	-	21642	300	317	328

ami49	336 2-pin nets					multi	time (s)		
	HPWL	ExactWL	# detour	(ExactWL-HPWL)/Exact(%)	(ExactWL-NP)/Exact(%)	HPWL	VG	LUT	Speedup
HPWL	622326	660007	82	5.71	-	287703	31		
NP	612749	639189	58	4.14	0.96	267709	180		
VG& LUT& S	586771	610775	63	3.93	-	268939	1081	731	734

n100	787 2-pin nets					multi	time (s)		
	HPWL	ExactWL	# detour	(ExactWL-HPWL)/Exact)(%)	(ExactWL-NP)/Exact)(%)	HPWL	VG	LUT	Speedup
HPWL	271605	276867	214	1.90	-	41315	91		
NP	273888	278441	195	1.64	0.76	42128	850		
VG& LUT& S	271637	275362	178	1.35	-	40784	5031	3149	2997



(a) The number of detours



(b) The length of detours

Figure 9 The number and length of detours using HPWL and LUT estimation in ami33 benchmark.

ratio increases, the error becomes larger.

5. Conclusions and Future Works

In [5], we discussed the theoretical results on the obstacle-aware estimation of the wire length of two-pin net. This paper implemented the algorithms of [5], and found that the preprocess takes a lot of time which can not be neglected. Thus, we proposed *Speedup Algorithm of LUT* which speeds up LUT Algorithm, but still consume over 20 times longer than HPWL method. Therefore, we proposed NP algorithm which can greatly improve the speed with minimum sacrifice in accuracy finally. With our proposed methods, the theoretical algorithm in [5] turns into a practical approach.

In this paper, we only discussed the case in which all the blocks are obstacles. It is easy to extend our method to handle the case in which a part of the blocks are opaque, by building the HVG based on those opaque blocks only.

Extension to handle multi-pin nets and integration of routing congestion should also be considered in the future.

References

[1] X. Zhang and Y. Kajitani, *"Space-planning: Placement of modules with controlled empty area by single-sequence"*, ASPDAC'04, pp. 25-30, 2004.

[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, *"VLSI module placement based on rectangle-packing by the sequence pair"*, IEEE TCAD, Vol.15, No. 12, pp. 1518-1524, 1996.

[3] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, *"B*-trees: a new representation for non-slicing floorplans"*, DAC'00, pp. 458-463, 2000.

[4] C. Kodama, K. Fujiyoshi, and T. Koga, *"T?A novel encoding method into sequence-pair"*, ISCAS'04, pp. 329-332, 2004.

[5] T. Yan, S. Li, Y.Takashima and H. Murata, *"A Theoretical Study on Wire Length Estimation Algorithms for Placement with Opaque Blocks"*, ASP-DAC' 07, pp. 268 - 273, 2007.

[6] J. S. B. Mitchell, *"Geometric shortest paths and network optimization"*, Handbook of Computational Geometry, Elsevier Science, pp. 633-702, 2000.