A Fast Longer Path Algorithm for Routing Grid with Obstacles using Biconnectivity based Length Upper Bound

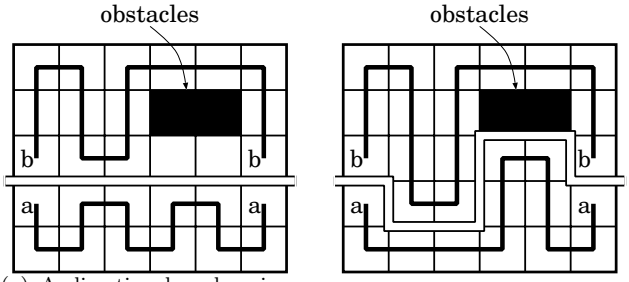
Yukihide Kohira Suguru Suehiro Atsushi Takahashi
Department of Communications and Integrated Systems, Tokyo Institute of Technology
2–12–1–S3–58 Ookayama, Meguro-ku, Tokyo, 152–8550 Japan
e-mail: {kohira,sgr,atsushi}@lab.ss.titech.ac.jp

Abstract— In recent VLSI systems, signal propagation delays are requested to achieve the specifications with very high accuracy. In order to meet the specifications, the routing of a net often needs to be detoured in order to increase the routing delay. A routing method should utilize a routing area with obstacles as much as possible in order to realize the specifications of nets simultaneously. In this paper, a fast longer path algorithm that generates a path of a net in routing grid so that the length is increased as much as possible is proposed. In the proposed algorithm, an upper bound for the length in which the structure of a routing area is taken into account is used. Experiments show that our algorithm utilizes a routing area with obstacles efficiently.

I. Introduction

In recent VLSI systems, due to the increase of operation frequency, signal propagation delays are requested to achieve the specifications with very high accuracy. In order to achieve the severe requirements, signal propagation delay is taken into account in the routing design of PCB (Printed Circuit Board). The specifications on routing design of PCB would be determined by the design of other parts. The signal propagation delay of a net consists of gate delay and routing delay, and depends on various parameters. However, in the routing design of PCB, the controllability of wire length is often focused on since it enables us to control the routing delay. If a routing method has the higher controllability on wire length, the routing delay would be controlled with higher accuracy. The length control is easy if the large area is reserved for route of a net. However, the routing area is usually limited, and shared by multiple nets. Therefore, a routing method should utilize the routing area with obstacles as much as possible in order to realize the specifications of multiple nets simultaneously. In the literatures, the problem formulations such as equal-delay or equal-length routings for multiple nets were often used to demonstrate the ability of methods.

For the routing problem for multiple nets, the methods in which a part of routing area is assigned to each net were proposed recently [1, 2]. In [1], a part of routing area is assigned to each net according to the direction of the net. In [2], a Lagrangian relaxation technique is used for length matching on multi-layers. These algorithms work well when the routing area does not have obstacles. However, the method proposed in [1] is not applicable to most PCB designs since obstacles are not taken into consideration. Although the method proposed in [2] is applicable when the routing area has obstacles, it is not guaranteed to obtain a feasible solution.



(a) A direction based assignment. (b) An optimal assignment.

Fig. 1. Routing area assignment.

Assume that net a and b are assigned routing areas as shown in Fig. 1 (a) and Fig. 1 (b), and a routing method tries to utilize the assigned routing area as much as possible. The sizes of the assigned areas for each net are the same in both figures, and the wire lengths of net a are the same in both figures. However, the wire lengths of net b differ. The existence of obstacles should be taken into account in routing area assignment.

The size of a routing area is not enough to evaluate the area. In order to evaluate a routing area with obstacles, the maximum wire length that can be achieved within the area should be known. However, it is an NP-hard problem since its decision version is NP-complete [3]. Therefore, a fast heuristic algorithm that evaluates a routing area with obstacles by the achievable wire length is desired.

In this paper, first, an upper bound for the wire length of a net in a routing grid with obstacles is proposed. The proposed upper bound uses the concept of the biconnected component to exclude useless areas for routing efficiently. Then, a longer path algorithm that generates the route of a net so that the wire length is increased as much as possible is proposed. In the proposed algorithm, the route of a net is greedily determined by using the proposed upper bound. In experiments, we show that a routing grid with obstacles is highly utilized by our proposed algorithm, and that the proposed upper bound helps our proposed

978-1-4244-2749-9/09/$25.00 ©2009 IEEE

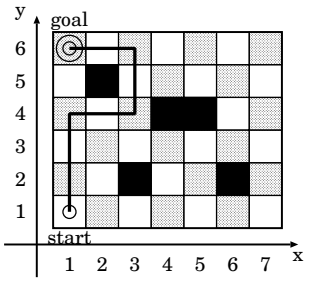
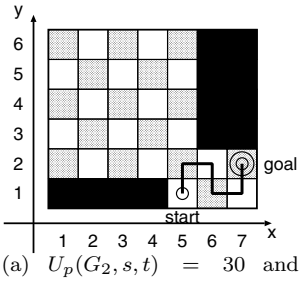
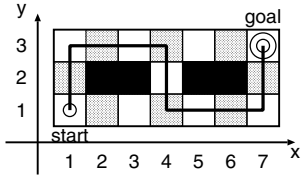


Fig. 2. Routing grid G_1 and a route ($L = 10$).



(a) $U_p(G_2, s, t) = 30$ and $L_{\max}(G_2, s, t) = 6$.



(a) $U_p(G_3, s, t) = 17$ and $L_{\max}(G_3, s, t) = 13$.

Fig. 3. Routing grid G_2 and G_3.

algorithm in generating a longer route. To the best of our knowledge, there is no good heuristic algorithm in comparison. Several heuristic algorithms are also introduced and used in comparison.

II. Preliminaries

In this paper, a routing area with single layer is represented by a routing grid consisting of unit squares, and a route is represented by thick line consisting of horizontal and vertical line segments as shown in Fig. 2. In the following, a unit square that corresponds to an obstacle is drawn by black in figures, and called as *off-square*. The other grids are drawn by white or gray and called as *on-squares*. From the on-squares, two squares are specified as a start and a goal of a route of a net. A route connects the start square and the goal square by line segments. A route passes each on-square at most once and must not pass any off-squares. A start square and a goal square are denoted by a single circle and a double circle in figures, respectively. The *length* of a route is the number of squares of the route including its start and goal squares.

A planar routing area is represented by G for simplicity. Note that an arbitrary routing area can be represented by setting obstacles on the boundaries. Each square of a routing grid is referred by their coordinate. The coordinate of the leftmost-bottom square is defined as $(1, 1)$. In the following, graph terminologies are used in explanation by assuming a grid graph is defined from a routing grid, but a routing grid representation is used in figure for simplicity and readability.

Let $L_{\max}(G, s, t)$ be the length of a longest route which connects start square s and goal square t by using on-squares in G only. If it is not confusing, $L_{\max}(G, s, t)$ is denoted by $L_{\max}$ and the length of a route is denoted by L for simplicity. An example is shown in Fig. 2. The start and goal squares are $(1, 1)$ and $(1, 6)$, respectively. The length of the route is 10 ($L = 10$).

The problem we concern is to find a route of a single net with $L_{\max}$. Methods for this problem is used as a subroutine of routing system or used to evaluate the area that will be assigned to a net. The problem to find a route with the minimum length in a grid graph with obstacles can be solved by maze routing in polynomial time [4]. On the other hand, it is known that Hamiltonian path problem in a grid graph with obstacles is NP-complete [3]. Since Hamiltonian path problem is a decision version of the problem to find a route with $L_{\max}$, the problem to find a route with $L_{\max}$ is NP-hard. Therefore, we investigate heuristic algorithms for this problem and upper bounds of length of a route.

III. Upper Bounds of Length

In this section, we introduce three upper bounds of length.

A. Upper Bound by Bipartition

Since a grid graph is bipartite, the set of squares is divided into two subsets, white and gray, so that any two squares in a subset are not adjacent to each other (see Fig. 2). Then, any route passes white and gray squares alternatively. Therefore, some on-squares of a subset can not be used in a route if the on-squares in the other subset are fewer than the on-squares in the subset. This analysis gives an upper bound of length of a route [5]. We assume that there is a path from a start square to a goal square.

Definition 1 *Let G be a grid graph, and s and t be squares in G. Let V_w (V_g) be the set of white (gray) on-squares in G. Let $U_p(G, s, t)$ be*

$$U_p(G, s, t) = \begin{cases} 2\min\{|V_w| - 1, |V_g|\} + 1 & (\textit{if } s, t \in V_w) \\ 2\min\{|V_w|, |V_g| - 1\} + 1 & (\textit{if } s, t \in V_g) \\ 2\min\{|V_w|, |V_g|\} & (\textit{otherwise}). \end{cases}$$

$U_p(G, s, t)$ is an upper bound of length of a route that connects s and t in G [5]. $U_p(G, s, t)$ is denoted by U_p for simplicity. The upper bound U_p is easy to calculate, but not tight in general. Examples in which U_p is larger than $L_{\max}$ are shown in Fig. 3.

B. Upper Bound by Biconnected Components

In this section, an upper bound in which the structure of a routing grid is taken into account is introduced. By using the concept of biconnected component, a tighter upper bound is obtained efficiently.

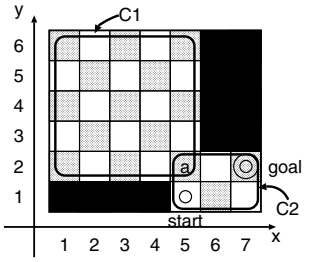


Fig. 4. Biconnected components of G_2. $U_c(G_2, s, t) = 6$.

A graph is said to be *biconnected* if it is connected even if any one vertex is removed from it. A *biconnected component* of a graph is a maximal biconnected subgraph of the graph. A vertex of a connected graph is said to be a *cut vertex* if the graph obtained by removing the vertex is unconnected. In general, the graph has at least one biconnected component. Every cut vertex belongs to at least two biconnected components, and the others belong to exactly one biconnected component. It is known that the time complexity to enumerate biconnected components is linear to the sum of the number of vertices and edges [6]. For example, the number of biconnected components of G_2 shown in Fig. 3 (a) is two, and biconnected components are shown in Fig. 4.

The sequence of biconnected components that corresponds to a route from the start square to the goal square is unique. A square that does not belong to a biconnected component in the sequence can not be used by any route from the start square to the goal square. This analysis gives an upper bound of length of a route.

Definition 2 *Let G be a grid graph, and s and t be squares in G. Let $(C_1, C_2, \ldots, C_n)$ be the sequence of biconnected components, and $(c_0, c_1, \ldots, c_n)$ be the sequence of cut vertices, such that $c_i \in C_i \cap C_{i+1}$ $(0 < i < n)$, $c_0 = s$, and $c_n = t$. Let $U_c(G, s, t)$ be*

$$U_c(G, s, t) = \begin{cases} 0 & (\textit{if } G \textit{ has no path between } s \textit{ and } t) \\ 1 & (\textit{if } s = t) \\ \sum_{i=1}^{n} U_p(C_i, c_{i-1}, c_i) - n & \\ & (\textit{otherwise}). \end{cases}$$

$U_c(G, s, t)$ is an upper bound of length of a route which connects s and t in G.

For example, $U_c(G_2, s, t) = 6$ which is equal to $L_{\max}(G_2, s, t)$ while $U_p(G_2, s, t) = 30$ (see Fig. 4). On the other hand, $U_c(G_3, s, t) = 17$ which is not equal to $L_{\max}(G_3, s, t) = 13$.

C. Proposed Upper Bound

The gap between the upper bound U_c defined in the previous section and the actual maximum length $L_{\max}$ becomes small, but still quite large in some cases. The upper bound can be improved by adopting a lookahead technique. That is, when a partial route from the start square or the goal square is generated, the sum of the upper bound of length of the remaining and the length of the generated partial route is an upper bound of length with the constraint such that the partial route is fixed. If all the possibilities of partial routes are enumerated, the actual maximum length does not exceed the maximum of them. That is, the maximum of them gives an upper bound of length. A tighter upper bound is obtained if the degree of lookahead is increased. However, there is a trade-off between the accuracy and the computational time. As the degree of lookahead is increased, the number of combinations explodes and the computation time increases exponentially.

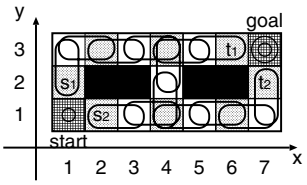


Fig. 5. Biconnected components of $G_3/\{s, t\}$. $U_a(G_3, s, t) = 13$.

In our proposed method, all the partial routes of length one from both a start square and a goal square are generated. Then, all the combinations are evaluated and the maximum of the obtained values is used as an upper bound of length by lookahead. $U_a(G, s, t)$ which is defined in the following is used as a tighter upper bound of length of a route.

Definition 3 *Let G be a grid graph, and s and t be squares in G. Let $A(v)$ be the set of on-squares which adjacent to v in G, and $G/\{s, t\}$ be the grid graph obtained from G by removing s and t. Let $U_a(G, s, t)$ be*

$$U_a(G, s, t) = \begin{cases} 0 & (\textit{if } G \textit{ has no path between } s \textit{ and } t) \\ 1 & (\textit{if } s = t) \\ 2 + \max_{s_n \in A(s), t_n \in A(t)} U_c(G/\{s, t\}, s_n, t_n) & \\ & (\textit{otherwise}). \end{cases}$$

For example, since upper bound $U_c(G_3/\{s, t\}, s_1, t_2) = 11$ and it is maximum among all combinations, $U_a(G_3, s, t) = 13$ (see Fig. 5).

Here, we discuss the time complexity in computing U_c and U_a. Let V be the set of squares in a grid graph G, and let E be the set of edge in G. Since $|E|$ is $O(|V|)$ in G, the time complexity of obtaining biconnected components is $O(|V|)$ [6]. Therefore, the time complexity in computing U_c is $O(|V|)$. Since the number of neighbors of a square is at most four, the time complexity in computing U_a is also $O(|V|)$.

IV. Proposed Routing Method

In this section, we propose a fast longer path algorithm that generates a route of a net so that the length of the

route is increased as much as possible. The proposed algorithm uses the upper bound U_a as a preferred function. We call our proposed routing method *US routing* (Upper bound based Seed routing).

In US routing, the route is determined by moving the *frontier* from the start square to the goal square. The frontier on a square is moved to an adjacent on-square such that U_a after the movement is maximum.

More precisely, the movement of the frontier in finding a route from s to t in G is described in the following. Assume that the frontier is on v. Let G' be the grid obtained from G by changing squares on the partial route from s to v defined by the movement of the frontier to off-square. Let $A(v)$ be the set of on-squares which are adjacent to v in G'. Let u be a square in $A(v)$ such that $U_a(G', u, t)$ is maximum among squares in $A(v)$. Then the frontier on v is moved to u in US routing.

Ties are broken by selecting a square such that the distance from the goal square t in G' is maximum, then broken arbitrarily. This tie breaking is used as preferred function in Furthest routing described in the next section.

US routing finds a route from a source square to a goal square if it exists. If the movement of the frontier forces a route not to reach the goal square, then U_a corresponding to the movement is 0. Such movement is never adopted if a route from a source square to a goal square exists since a movement whose U_a is positive always exists.

Here, we discuss the time complexity of US routing. The frontier arrives at the goal square by at most $|V|$ movements. In each movement, the number of computation of the upper bound U_a is at most four. Therefore, since the time complexity of U_a is $O(|V|)$, the time complexity of US routing is $O(|V|^2)$.

V. Routing Methods for Comparisons

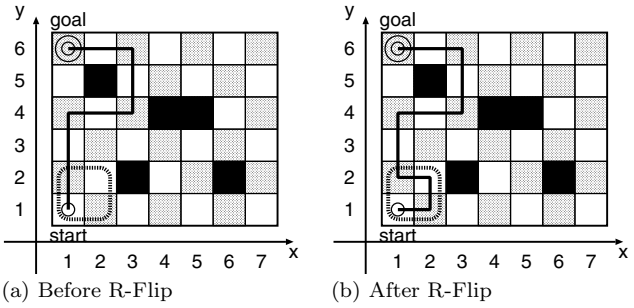
In this section, several heuristic algorithms are introduced to use them in comparison. Algorithms consist of two stages. At the first stage which is called *seed stage*, a route which connects the start and goal squares is generated. At the second stage which is called *expansion stage*, a route obtained by the first stage is modified greedily to increase the length of the route.

A. Seed Stage

In the seed stage, a route which connects the start and goal squares is generated by using one of three routing methods, "US", "Maze", or "Furthest". US routing was explained in the previous section. The other two routing methods are explained briefly.

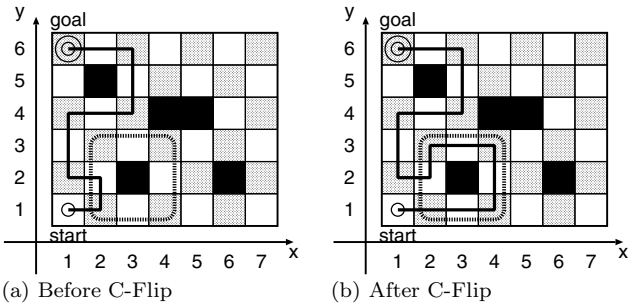
In Maze routing, the frontier is repeatedly moved to an adjacent on-square such that the distance from the goal square is minimum. The distance from the goal square is the length of a shortest path from the goal square using on-squares only. In Maze routing, first, each square is assigned a distance label, and the frontier moves according to the distance label without updating distance labels. It is well-known that a route with the minimum length is obtained by Maze routing and that the time complexity of Maze routing is $O(|V|)$ [4].

In Furthest routing, the frontier is repeatedly moved to an adjacent on-square such that the distance from the goal square is maximum. In contrast with Maze routing, distance labels are updated after each movement of the frontier since the distance from the goal square might be changed by a movement of the frontier. Of course, a route with the maximum length is not necessarily obtained by Furthest routing. The time complexity of Furthest routing is $O(|V|^2)$, since the time complexity of updating the distance label is $O(|V|)$ and the number of movements of the frontier is at most $|V|$.



(a) Before R-Flip (b) After R-Flip

Fig. 6. R-Flip.



(a) Before C-Flip (b) After C-Flip

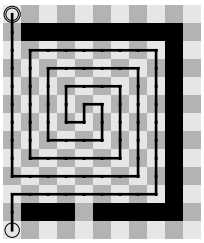
Fig. 7. C-Flip.

B. Expansion Stage

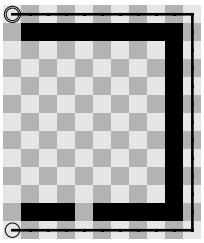
In the expansion stage, a route obtained by the seed stage is modified greedily to increase the length of the route. *R-Flip* and *C-Flip* are introduced as the methods used in the expansion stage.

In R-Flip, a partial route of length two is detoured by using two adjacent on-squares (see Fig. 6). That is, a partial route of length two is replaced by a partial route of length four. In R-Flip, a partial route in four squares in a 2 times 2 rectangle such that a partial route passes two of them and that the others are on-square is modified. In our implementation, R-flip is applied greedily by searching an appropriate rectangle along the route from the start square to the goal square.

C-Flip is a generalization of R-Flip. A partial route R in the route is replaced by another partial route R' that passes on-squares and that has the same terminals

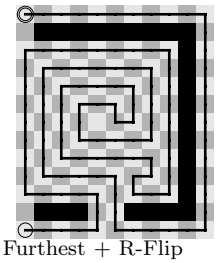


(a) Maze + R-Flip ($L = 85$).

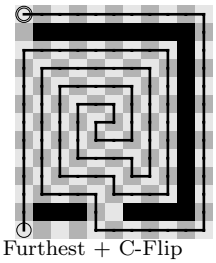


(b) Furthest + R-Flip ($L = 33$).

Fig. 8. Routes for Data7.



(a) Furthest + R-Flip ($L = 115$) (optimal).



(b) Furthest + C-Flip ($L = 113$).

Fig. 9. Routes for Data8

of R. C-Flip is applied if the length of R is larger than the length of R' so that the length of the route increases (see Fig. 7). To find a candidate of C-Flip, each square is assigned a label that corresponds to the length of a shortest path from a square v on the route that passes on-squares. If the label of a square on the route is larger than the length of a partial route from v, C-Flip is applied. In our implementation, such pair is searched by changing the start square v of a partial route from the start square.

In [7], Flip is proposed as a route modification method. Flip modifies the route according to a face on the plain graph. Since the rectangle with width two and length two is a face, R-Flip is a kind of Flip. On the other hand, C-Flip might not be a kind of Flip since the cycle defined by C-Flip might not be a face. C-Flip is introduced as a specialized method for increase of the length of a route.

VI. Experimental Results

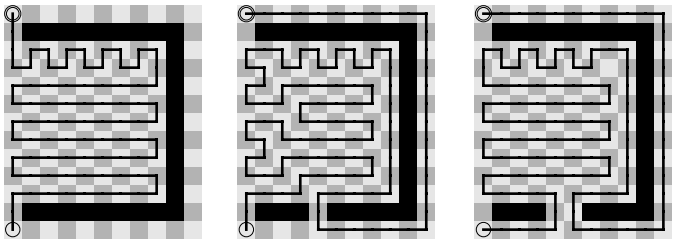
We implement the computation of upper bounds and the routing methods in C++, which compiled gcc4.1.2, and executed on a PC with 2.93GHz Intel Core 2 CPU and 2GB RAM. We apply each method to 10 sample data.

The experimental results are shown in Table. I. Examples of obtained routes are shown from Fig. 8 to Fig. 13.

Every upper bound is less than the number of on-grids for every sample data except Data10. From Data1 to Data8, $U_c = U_p$ since the number of biconnected components is one. For Data9 and Data10, $U_c < U_p$ since the dead ends are excluded in U_c. From Data6 to Data8, $U_a < U_c$ since more than one biconnected component is generated by lookahead.

The gap between the length of route obtained by Maze routing or Furthest routing and the proposed upper bound U_a is large, since Maze routing and Furthest routing are not appropriate for the maximum length problem. Although the length of the route is improved at the expansion stage, the gap is still large in most data. The best combination of methods of seed stage and expansion stage depends on data. In average, Furthest routing is better than Maze routing as the method of seed stage and C-Flip is better than that R-Flip as the method of expansion stage. Examples of routes obtained by Maze routing or Furthest routing are shown in Fig. 8 and Fig. 9.

While, US routing obtains a longer route such that the route is not improved at expansion stage in most cases



(a) Data6 ($L = 85$). (b) Data7 ($L = 113$). (c) Data8 ($L = 115$).

Fig. 11. Longest routes by US routing (optimal).

(see from Fig. 10 to Fig. 13). Moreover, US routing is equal to or close to the proposed upper bound U_a. This fact implies that the proposed upper bound U_a and US routing are promising.

Since US routing is heuristic, the length of the route obtained by US routing is less than that by other routing methods in a few case (see Fig. 13). The computation time is long since the number of computations of biconnected components is large. These issues remain as future works.

VII. Conclusions

In this paper, we propose an upper bound for the wire length of a net in a routing grid with obstacles based on the biconnected component and a longer path algorithm US routing which greedily determines the route by using the proposed upper bound. In experiment, the lengths of routes obtained by US routing are longer than those by compared methods in most data, and the upper bound is same or close to the wire length obtained by US routing. These results mean the proposed upper bound and US routing are promising.

The computation time of US routing is large since it computes our proposed upper bound iteratively. Improvement of US routing in terms of the computation time will be in our future works. Moreover, the development of a routing method for multiple nets is in our future works, in which the concept of US routing will be used.

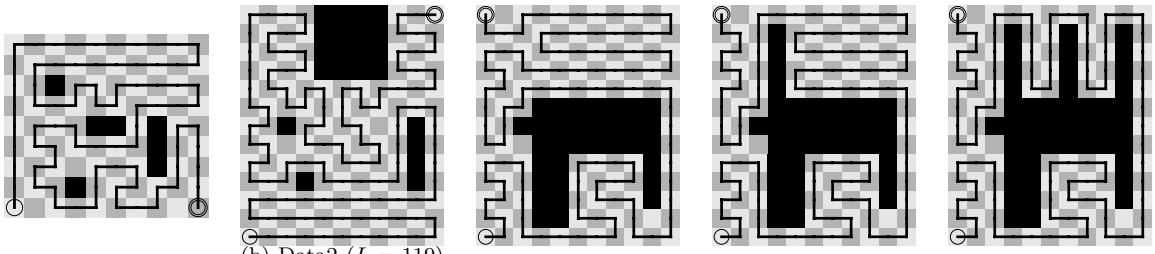
Acknowledgements

The authors would like to thank Toshiyuki Shibuya of Fujitsu Laboratories of America, Inc. for his helpful suggestions.

TABLE I RESULTS.

	area	Upper bound				Maze routing			Furthest routing			US routing		
		#on	U_p [5]	U_c	U_a		+R-F.	+C-F.		+R-F.	+C-F.		+R-F.	+C-F.
Data1	10x9	83	82	82	82	10	78	80	78	80	80	80	—	—
Data2	11x13	121	**119**	**119**	**119**	23	**119**	**119**	105	115	115	**119**	—	—
Data3	11x13	110	109	109	109	13	77	107	105	107	107	107	—	—
Data4	11x13	106	105	105	105	13	37	103	99	103	103	103	—	—
Data5	11x13	98	97	97	97	13	37	95	91	95	95	95	—	—
Data6	11x13	116	115	115	**85**	13	**85**	33	33	—	—	**85**	—	—
Data7	11x13	117	115	115	**113**	13	85	**113**	33	—	**113**	**113**	—	—
Data8	11x13	118	117	117	**115**	13	89	113	33	**115**	113	**115**	—	—
Data9	20x20	297	291	267	267	9	—	231	251	—	261	251	—	—
([s])												0.01	0.01	0.01
Data10	70x100	6654	6654	6650	6650	154	6584	6594	6490	6566	6566	6626	6628	6628
([s])									1.21	1.21	1.21	4.76	4.80	4.80

#on the number of on-squares
+R-F. the length of the route obtained by R-Flip after seed stage
+C-F. the length of the route obtained by C-Flip after seed stage
— means that the length is not increased by the expansion method.
bold figure means that the length of the route is equal to the upper bound.
* Computation time which is less than 0.01[s] is omitted.



(a) Data1 ($L = 80$). (b) Data2 ($L = 119$) (optimal). (c) Data3 ($L = 107$). (d) Data4 ($L = 103$). (e) Data5 ($L = 95$).

Fig. 10. Routes by US routing.

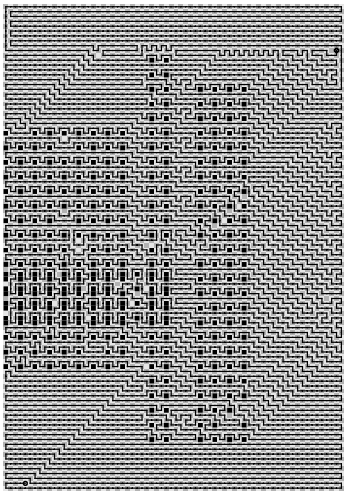
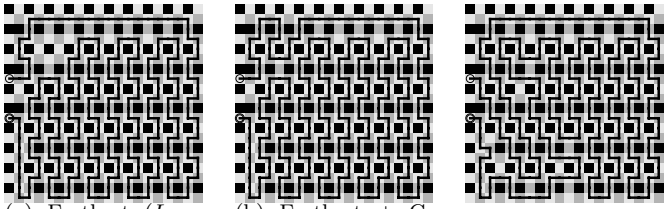


Fig. 12. Route for Data10 by US routing ($L = 6626$).



(a) Furthest ($L = 251$). (b) Furthest + C-Flip ($L = 261$). (c) US ($L = 251$).

Fig. 13. Routes for Data9.

REFERENCES

[1] M. M. Ozdal and M. D. F. Wong, "Algorithmic Study of Single-Layer Bus Routing for High-Speed Boards," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 3, pp. 490–503, 2006.

[2] M. M. Ozdal and M. D. F. Wong, "A Length-Matching Routing Algorithm for High-Performance Printed Circuit Boards," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2784–2794, 2006.

[3] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamiltonian Paths in Grid Graphs," *SIAM, Comp.*, vol. 11, no. 4, pp. 676–686, 1982.

[4] C. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electronic Comput.*, no. EC-10, pp. 346–365, 1961.

[5] S. Suehiro, Y. Kohira, and A. Takahashi, "An Estimation of Maximum Wire Length in Routing Area with Obstacles," *Technical Report of IEICE, CAS2007–97*, pp. 19–23, 2008. (In Japanese).

[6] R. E. Tarjan, "Depth-First Search and Linear Graph Algorithms," *SIAM, Comp.*, vol. 1, no. 2, pp. 146–160, 1972.

[7] Y. Kubo, Y. Takashima, S. Nakatake, and Y. Kajitani, "Self-reforming Steiner Tree by Flip and Applications to VLSI Interconnection," *IPSJ Journal*, vol. 41, no. 4, pp. 881–888, 2000. (In Japanese).