論文 / 著書情報
Article / Book Information

Title	A Length Matching Routing Algorithm for Set-Pair Routing Problem
Authors	Yuta Nakatani, Atsushi Takahashi
出典 / Citation	IEICE Trans. Fundamentals, Vol. E98-A, No. 12, pp. 2565-2571
発行日 / Pub. date	2015, 12
URL	http://search.ieice.org/
権利情報 / Copyright	本著作物の著作権は電子情報通信学会に帰属します。 Copyright (c) 2015 Institute of Electronics, Information and Communication Engineers.

PAPER Special Section on VLSI Design and CAD Algorithms

A Length Matching Routing Algorithm for Set-Pair Routing Problem

Yuta NAKATANI[†a)], *Nonmember and* **Atsushi TAKAHASHI**[†b)], *Senior Member*

SUMMARY In the routing design of interposer and etc., the combination of a pin pair to be connected by wire is often flexible, and the reductions of the total wire length and the length difference are pursued to keep the circuit performance. Even though the total wire length can be minimized by finding a minimum cost maximum flow in set pair routing problems, the length difference is often large, and the reduction of it is not easy. In this paper, an algorithm that reduces the length difference while keeping the total wire length small is proposed. In the proposed algorithm, an initial routing first obtained by a minimum cost maximum flow. Then it is modified to reduce the maximum length while keeping the minimum total wire length, and a connection of the minimum length is detoured to reduce the length difference. The effectiveness of the proposed algorithm is confirmed by experiments.

key words: *set-pair routing, interposer, PCB, routing algorithm*

1. Introduction

In the routing design such as silicon interposer [1], [2], printed circuit board (PCB) and etc., the combination of a pin pair to be connected by wire is often flexible when a wire is required to connect passive elements, I/O pins of reconfigurable chip or etc. In such cases, whether the connection requirements can be achieved is easily checked by network flow algorithms. Moreover, the total wire length of connections can be minimized by finding a minimum cost maximum flow as used in [2]. However, the length difference of wires is often large even when the total wire length is minimized. It would cause the delay mismatch and it is not acceptable in recent high-speed interposer/PCB designs. In order to keep the circuit performance good enough, the length matching of wires is pursued while keeping the wire lengths small, but it is not easy even if there exists a flexibility on pin pairs.

In this paper, a length matching routing algorithm for a set-pair routing problem is proposed. In the set-pair routing problem, the connection requirements are given between the source-pin set and the sink-pin set. One pin from the source-pin set and one pin from the sink-pin set are connected by wire to propagate a signal so that no pin is shared by more than one signal.

An example of a set-pair routing problem instance is shown in Fig. 1 in which red rectangles represent source-pins and in which blue rectangles represent sink-pins. Two routing patterns are shown in Fig. 1(a) and (b). The total wire length of them is minimum. Here, the number of grid segments used is set to the length. A minimum cost maximum flow algorithm typically generates a routing pattern that has large length difference as shown in Fig. 1(a), even though there exist routing patterns that have small length difference as shown in Fig. 1(b).

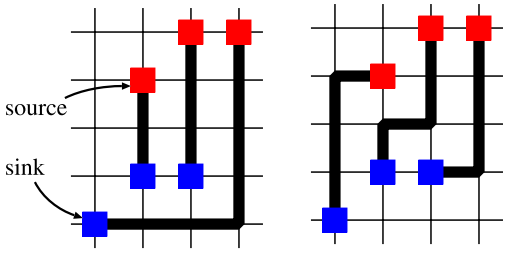


Fig. 1 Example of set-pair routing.

In the set-pair routing problem, the length matching is not easy even though the minimum total wire length is achieved in polynomial time. The length matching in the set-pair routing would be improved by applying length matching algorithms for ordinary routing problems [3]–[5]. However, the results seem not good enough since they generate routing patterns by utilizing characteristics of problem specifications well, but without utilizing the flexibility of pin pairs.

Our proposed length matching algorithm for a set-pair routing problem reduces the maximum wire length as well as the length difference as much as possible. Our algorithm utilizes network flow algorithms effectively, and consists of three stages. Our algorithm, first, finds a routing pattern that has the minimum total wire length. Second, the routing pattern is modified to improve the length matching by reducing the maximum wire length while keeping the total wire length minimum. Finally, the length matching is further improved by increasing the minimum wire length while increasing the total wire length. In the final stage, the increase of the total wire length is suppressed since the maximum length is reduced as much as possible in the previous stage.

Our length matching algorithm in this paper is a heuristic even though the time complexity of the problem that minimizes the length difference of wires is not apparent. The enhancements of our algorithm that take the delay matching into account as well as that take the target delay of each signal into account would be possible.

Manuscript received March 11, 2015.
Manuscript revised July 10, 2015.
[†]The authors are with Tokyo Institute of Technology, Tokyo, 152-8550 Japan.
a) E-mail: nakatani@eda.ce.titech.ac.jp
b) E-mail: atsushi@eda.ce.titech.ac.jp
DOI: 10.1587/transfun.E98.A.2565

The rest of paper is organized as follows. Definitions and basic notations especially for network flow are given in Sect. 2. The complexities of several routing problems including the set-pair routing problem are discussed in Sect. 3. In Sect. 4, our proposed length matching algorithm is introduced. Then, the experimental results in which our algorithm is applied to several types of pin distributions on a planar grid with obstacles are given in Sect. 5. Finally, the conclusion is given in Sect. 6.

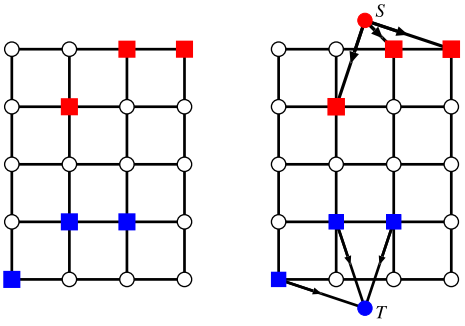
2. Preliminaries

The routing area is modeled as an undirected graph, called *routing graph*, where a vertex corresponds to a grid point of the routing area and an edge corresponds to a candidate of wire segment. Source-pins and sink-pins are placed on grid points of a routing graph. The routing graph also represents a flow graph [6]–[8] in which each vertex v has capacity $c(v)$ and weight $w(v)$. Source-pins and sink-pins are sources and sinks of the flow graph, respectively.

In a flow graph, the primary source S that is connected to source-pins and the primary sink T that is connected from sink-pins are added to convert the flow graph to a single source single sink flow graph. The capacity and weight of both the primary source and the primary sink are infinite and zero, respectively.

For example, in Fig. 2(a), the routing graph of the set-pair routing problem instance shown in Fig. 1 is shown. In Fig. 2(b), the flow graph that corresponds to the routing graph shown in Fig. 2(a) is shown.

In our implementation, a vertex capacity flow graph is converted to an equivalent directed flow graph with edge capacity. The conversion from a vertex capacity to an edge capacity is explained in the following. A vertex v except the primary source and the primary sink is converted to two vertices, named *in-vertex* v_i and *out-vertex* v_o, which is connected by a directed edge e_v from v_i to v_o, The directed edge e_v has capacity $c(v)$ and weight $w(v)$. An undirected edge in a vertex capacity graph is converted to two directed edges from out-vertex to in-vertex of the corresponding end vertices of the undirected edge. These edges have infinite capacity and zero weight.



(a) Routing Graph (b) Flow Graph

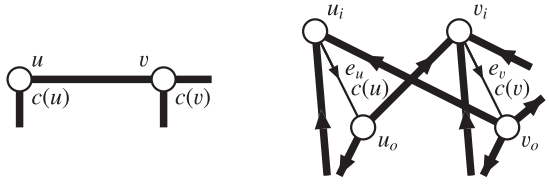
Fig. 2 Routing graph and flow graph.

For example, in Fig. 3, vertex u of capacity $c(u)$ and weight $w(u)$ is converted to in-vertex u_i, out-vertex u_o, and edge $e_u = (u_i, u_o)$ of capacity $c(u)$ and weight $w(u)$. An undirected edge (u, v) of a vertex capacity graph is converted to two directed edges (u_o, v_i) and (v_o, u_i).

In this paper, we assume that the vertex capacity $c(v)$ is 1 for any vertex v except the primary source and the primary sink, and that a flow consists of a set of streams of unit amount. Each stream proceeds along directed edges from the primary source to the primary sink, and corresponds to a wire between a source-pin and a sink-pin. A flow is valid when capacity constraints are satisfied. A flow is said to be maximum if the number of streams is maximum among valid flows. The vertex weight $w(v)$ can be set to arbitrary for any vertex, but is set to 1 in the explanation for simplicity. The cost of a flow is defined as the sum of cost of streams. The cost of a stream is the sum of edge weights each of which is proportional to the amount of the stream. A minimum cost maximum flow can be obtained in polynomial time if the cost of a flow is defined as above [9].

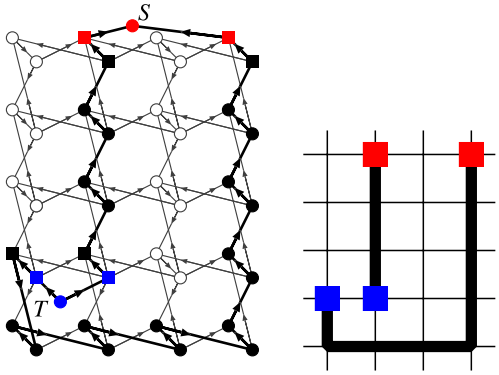
In order to obtain a minimum cost maximum flow, the *residue graph* of a valid flow is defined. The residue graph is obtained from the flow graph by reversing the direction of edges in which a stream flows. The sign of the weight of an edge is also reversed when its direction is reversed. It is known that a maximum flow is the minimum cost if and only if the residue graph contains no negative weight directed cycle.

For example, in Fig. 4(a), the residue graph of a flow graph corresponding to the routing pattern of the total wire



(a) Vertex capacity (b) Edge capacity

Fig. 3 Vertex capacity to edge capacity.



(a) Residue Graph (b) Routing (total=11)

Fig. 4 Residue graph of a flow.

length 11 shown in Fig. 4(b) is shown. Note that the total wire length is not minimum though it is obtained by a maximum flow algorithm. In Fig. 4(a), thick black edges correspond to edges in which a stream flows.

3. Length Matching of Routing

3.1 Ordinary Routing Problem

In ordinary routing problems, a set of pins to be connected by wire is given as a net. The problem finding a wire with the minimum length for a single net is formulated as a shortest path problem on a weighted graph or a Steiner tree problem on a weighted graph. When the number of pins of a net is two, a shortest path of the net can be obtained by polynomial time algorithms when the weight of an edge of the routing graph is non-negative [7], [10]. While the problem finding a minimum Steiner tree that connects three or more pins is NP-hard in general, and it is considered that it is impossible to find an optimal solution in polynomial time [9], and various heuristic algorithms have been proposed so far.

Furthermore, in ordinary routing problems, multiple nets are given in general. It is difficult for multiple nets to determine whether all the connection requirements can be achieved. As it is well-known, the problem finding disjoint paths is NP-hard when two or more two-pin nets are given on a planar graph [9]. It is also known that the problem is still NP-hard even if the graph is restricted to a grid graph [11].

Network flow is often used to solve a routing problem. However most of them are impractical except the situations in which a flow always corresponds to the connection of a net [5]. Multi commodity flow can be used to formulated a routing problem, but it is not effective in general since the problem is NP-hard if the amount of flow is restricted to an integer even if the number of commodities is two [12].

So far, various kinds of algorithm have been proposed for various kinds of design objectives. For example, the length matching algorithms that control the length of wire were proposed for ordinary routing problems under several restrictions were proposed [3]–[5].

3.2 Set-Pair Routing Problem

In the routing design such as silicon interposer [1], [2], printed circuit board and etc., the combination of a pin pair to be connected by wire is often flexible when a wire is required to connect passive elements, I/O pins of reconfigurable chip or etc. In such cases, the number of signals to be propagated and the locations of pins are given as a problem instance specification.

Set-Pair Routing Problem is defined as follows.

Set-Pair Routing Problem :

The connection requirements are given between two sets of pins, named source-pin set and sink-pin set. The locations of source-pins and sink-pins are given as an input. One pin from the former and the other pin from the latter are requested to be connected in the given routing area without intersecting each other.

Objectives

- The reduction of the total wire length
- The reduction of the maximum wire length
- The reduction of the wire length difference
- The achievement of the target wire length

A various types of set-pair routing problems can be defined depending on the objectives derived from applications. A basic set-pair routing problem can be regarded as a disjoint path problem between two vertices. In the problem formulation as the disjoint path problem, the primary source that is connected to source-pins and the primary sink that is connected from sink-pins are defined and are requested to be connected by internally disjoint paths. Two internally disjoint paths that connect a given pin pair can be obtained in a linear time [13]. Even though the time complexity of the problem that minimizes the length difference of wires is not apparent, the problem of finding disjoint paths of the designated length or length upper bound is NP-hard in general [14].

In a basic set-pair routing problem, whether the connection requirements can be achieved is easily checked by network flow algorithms. Also, a set of wires that achieves the minimum total wire length can be obtained by obtaining a minimum cost maximum flow which can be obtained by a polynomial time [2], [9]. However, it is not easy to achieve the length matching of wires even if there exists a flexibility on pin pairs. The difficulties in set-pair routing problems are in the reduction of the maximum wire length of connections and the length mismatch among wires.

4. Length Matching Routing Algorithm for Set-Pair Routing

4.1 Overview

Our proposed length matching algorithm for a set-pair routing problem reduces the maximum wire length as well as the length difference as much as possible. Our algorithm utilizes network flow and consists of three stages. The outline of our algorithm is shown in Fig. 5.

Our algorithm, first, finds a routing pattern by using a minimum cost maximum flow algorithm. The total wire length of the routing is minimum. However the length difference is tend to be large in general. Second, the initial routing is modified to improve the length matching by reducing the maximum wire length while keeping the total wire length minimum. Pin pairs are exchanged to reduce the maximum wire length. Finally, the length matching is further improved by increasing the minimum wire length while increasing the total wire length. In the final stage, the increase of the total wire length is suppressed since the maximum length is reduced as much as possible in the previous stage.

Stage 1 (Total Reduction)
Obtain a routing pattern that has the minimum total wire length by finding a minimum cost maximum flow of the corresponding flow graph.
Stage 2 (Maximum Reduction)
Reduce the maximum wire length while keeping the total wire length minimum by greedily modifying a routing pattern so that an earlier length sequence is obtained.
Stage 3 (Minimum Increase)
Lengthen a wire of the minimum length as much as possible by using R-Flip iteratively while the maximum length is kept.

Fig. 5 Proposed length matching algorithm.

Our length matching algorithm in this paper is a heuristic even though the time complexity of the problem that minimizes the length difference of wires is not apparent. The achievement of the target wire length is not taken into account in our algorithm. But it is not difficult to enhance our algorithm to take it into account. The details of our proposed algorithm are described in the following subsections.

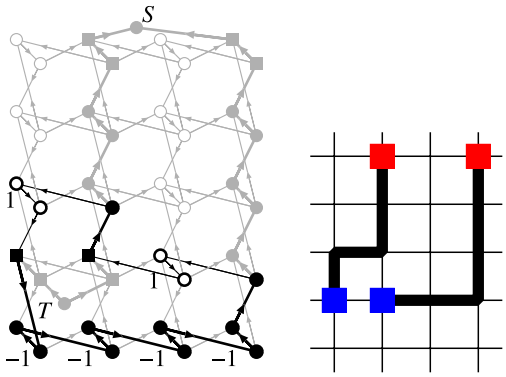
4.2 Total Wire Length Minimization

In order to obtain a routing whose total wire length is minimum, a minimum cost maximum flow algorithm is used. It is known that a minimum cost maximum flow can be obtained in polynomial time [2], [9].

In our implementation, in order to obtain a maximum flow, a flow stream of a unit amount from primary source to primary sink is iteratively added by finding a augmenting path in the flow graph iteratively until there is no augmenting path. Augmenting paths are obtained by using a breadth-first-search base algorithm. After a maximum flow is obtained, the cost of a maximum flow is reduced by modifying the flow iteratively by adding a closed stream that corresponds to a negative weight directed cycle of the residue graph of the flow until there is no negative weight directed cycle. The cost of a maximum flow is reduced to which a closed stream that corresponds to a negative weight cycle in the residue graph is added. A negative weight cycle in a weighted directed graph is obtained by using the negative cycle detection algorithm shown in [15] which is based on Bellman-Ford shortest path algorithm [16].

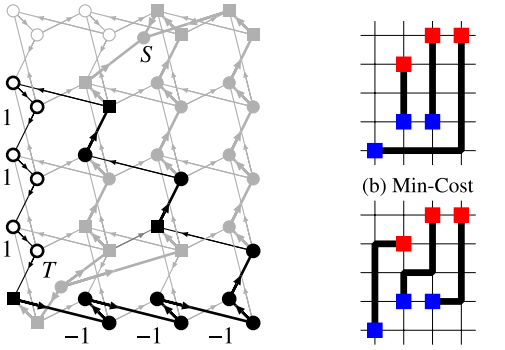
In Fig. 6(a), a negative weight cycle of weight -2 in the residue graph shown in Fig. 4(a) is drawn in black. In Fig. 6(b), the routing pattern of the total wire length 9 corresponding to the flow graph obtained by adding a closed stream on the cycle is shown. Note that the change of the total wire length is equal to the weight of the cycle.

The total wire length of the routing that corresponds to the obtained minimum cost maximum flow is minimum. However the length difference is tend to be large since the routing is generated by using breadth-first-search base algorithm. Shorter wires that connects near pins are generated in earlier and the remaining distant pins are connected later. Even though connections are modified to reduce the total wire length, the length difference often remains large. It is often the case that the total wire length is minimum but the length difference is large. The minimum cost maximum flow algorithm does not modify the routing if it achieves the minimum total wire length.



(a) Negative Cycle (weight=−2) (b) Routing (total=9)

Fig. 6 Negative weight cycle of residue graph.



(a) Zero Weight Cycle (b) Min-Cost (c) Length Matched

Fig. 7 Zero weight cycle of residue graph.

4.3 Maximum Wire Length Reduction

A routing pattern that has the minimum total wire length does not necessarily achieve the length matching. In order to improve the length matching, the maximum length of the routing pattern is reduced while keeping the total wire length minimum.

Typically, the length difference is tend to be large when a routing pattern is obtained by using a typical minimum cost maximum flow algorithm. The length matching is improved if a directed cycle in a residue graph that improves the length matching is obtained, but the characterization such cycles is not easy.

For example, in Fig. 7(a), the residue graph of the flow corresponding to the routing pattern shown in Fig. 7(b) is shown. The residue graph contains no negative weight directed cycle. In Fig. 7(a), the zero weight cycle that converts the routing pattern shown in Fig. 7(b) to the routing pattern shown in Fig. 7(c) is drawn in black.

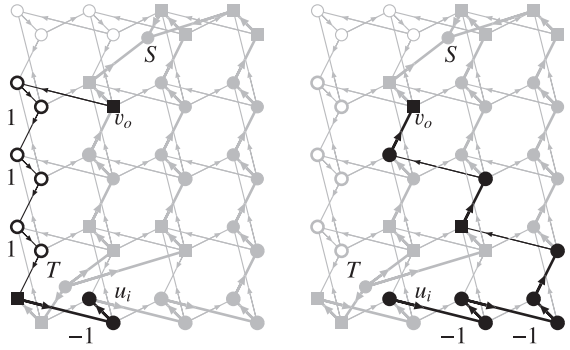
A zero weight directed cycle of the residue graph

would be found by a similar method to negative cycle detection algorithms. However, most of zero weight directed cycles do not improve the length matching. If pin pairs to be connected are not exchanged, the length matching is not improved if the minimum total length is maintained. A zero weight directed cycle interacts with at least one flow stream. If it interacts with only one flow stream, then pin pairs to be connected are not exchanged. In order to find a zero weight directed cycle that has potential to improve the length matching, the cycle must interact with at least two flow streams so that pin pairs to be connected are exchanged. In order to find such a zero weight directed cycle, our procedure checks whether there exists a zero weight directed cycle that contains two vertices on the different flow streams.

In our procedure in this stage, first, two vertices v and u are chosen from grid points on the different flow streams. Second, a shortest path from out-vertex v_o of v to in-vertex u_i of u in the residue graph is obtained by using a Bellman-Ford shortest path base algorithm. Then, a shortest path from u_i to v_o that excludes internal vertices of the shortest path from v_o to u_i is obtained to form a directed cycle when two paths are concatenated. If the sum of weights of two paths is zero, then a zero weight directed cycle which is the concatenation of two paths is found. The current routing pattern is modified if the length matching is improved when the found zero weight directed cycle is applied. Otherwise the current routing pattern is kept. This is repeated until no better routing pattern is found for combinations of vertices.

In order to find a cycle that has a potential to improve the length matching, at least one grid point is chosen from a longer flow stream. Even though the existence check of a zero weighted directed cycle is a heuristic, and there might be missed a cycle in terms of chosen two vertices, a zero weighted directed cycle would be found for several combinations of vertices.

For example, in Fig. 8, two shortest paths between v_o and u_i in the residue graph shown in Fig. 7(a) are drawn in black. The weights of the shortest paths from v_o to u_i and from u_i to v_o are 2 and -2, respectively. The concatenation of these two paths forms the cycle shown in Fig. 7(a).



(a) v_o to u_i (weight=2) (b) u_i to v_o (weight=−2)

Fig. 8 Shortest paths between v_o and u_i.

In our implementation, the order of pairs of vertices to find a candidate zero weight directed cycle is defined as follows. The order of pairs of vertices is defined for each pair of wires as follows. For each vertex chosen along the longer wire from sink pin to source pin, a vertex to be paired is chosen along the shorter wire from source pin to sink pin. The order of pairs of wires to generate pairs of vertices is defined as follows. For each wire in decreasing order of wire length, a shorter distinct wire to be paired is selected in increasing order of wire length (ties are broken arbitrary).

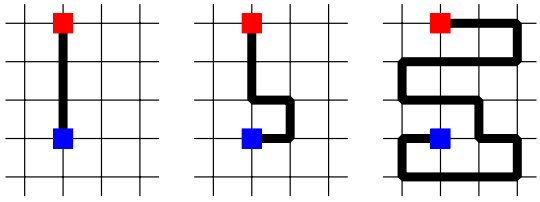
In our procedure, the current routing pattern is modified if a better routing pattern is obtained. If a routing pattern is evaluated by using the maximum wire length or by using the length difference, the evaluation is often the same even if the routing pattern looks improved. In order to improve the search ability while keeping the convergence of modifications, the length sequence of a routing pattern is used to evaluate the routing pattern.

The *length sequence* of a routing pattern is defined as a decreasing order of lengths of wires in the routing pattern. In the lexicographical order of the length sequence of routing patterns, a routing pattern that has a smaller maximum wire length is earlier than that has a larger maximum wire length. If the maximum wire lengths are the same, the smaller second largest wire length is earlier, and so on. Even though an earlier lexicographical order does not necessarily mean a smaller length difference, the length difference of a routing pattern is tend to be small if it has an earlier lexicographical order.

4.4 Minimum Wire Length Increase

The reduction of the wire length difference can be also achieved by lengthening shortest wires instead of the reduction of the maximum wire length. In the final stage, in order further improve the length matching the length of a shortest wire is increased by applying simple detour modification called R-flip in [17] while keeping the maximum wire length. The total wire length is increased in this stage, but the increase of the total wire length is suppressed since the maximum length is reduced as much as possible in the previous stage.

In our procedure, R-flips are applied iterative to an arbitrary wire segment of an arbitrary shortest wire, even though the final result depends on how R-flips are applied. In our implementation, a feasible R-flip which is found first



(a) Initial (b) R-flip (c) Final

Fig. 9 R-flip.

is applied. In order to find a feasible R-flip, the feasibility of R-flips to upper, lower, left, and right directions for each wire segment from sink pin to source pin along an arbitrary shortest wire are checked in this order.

For example, R-flip is applied to the routing pattern shown in Fig. 9(a), the routing pattern shown in Fig. 9(b) is obtained. By applying R-flip repeatedly, the routing pattern shown in Fig. 9(c) is obtained.

5. Experiments

The proposed length matching algorithm is implemented by C++ programming language on Intel(R) Core i3-550 CPU 3.20 GHz, 4 GB Memory.

The problem instances are generated randomly to fit with several situations as single layer routing problem. The number of source-pins is equal to the number of the sink-pins. Obstacles are randomly inserted in routing area. In E series instances, source pins are on array and sink pins are on the boundary. In B series instances, source pins and sink pins are generated to align on lines. In S series instances, source pins and sink pins are generated in top half and bottom half of the single layer routing region. In F series instances, source pins and sink pins are on arrays of different size.

The results are shown in Table 1. The statistics of the output of each stage is shown. The number of nets, the number of grids to which an obstacle is placed are shown in "#Net" and "#Obst", respectively. The total wire length, the maximum wire length, and the wire length difference are shown in L_{tot}, L_{max}, and d_{max}, respectively. The computation time (sec.) of each stage is also shown. Routing patterns obtained are shown in Fig. 10.

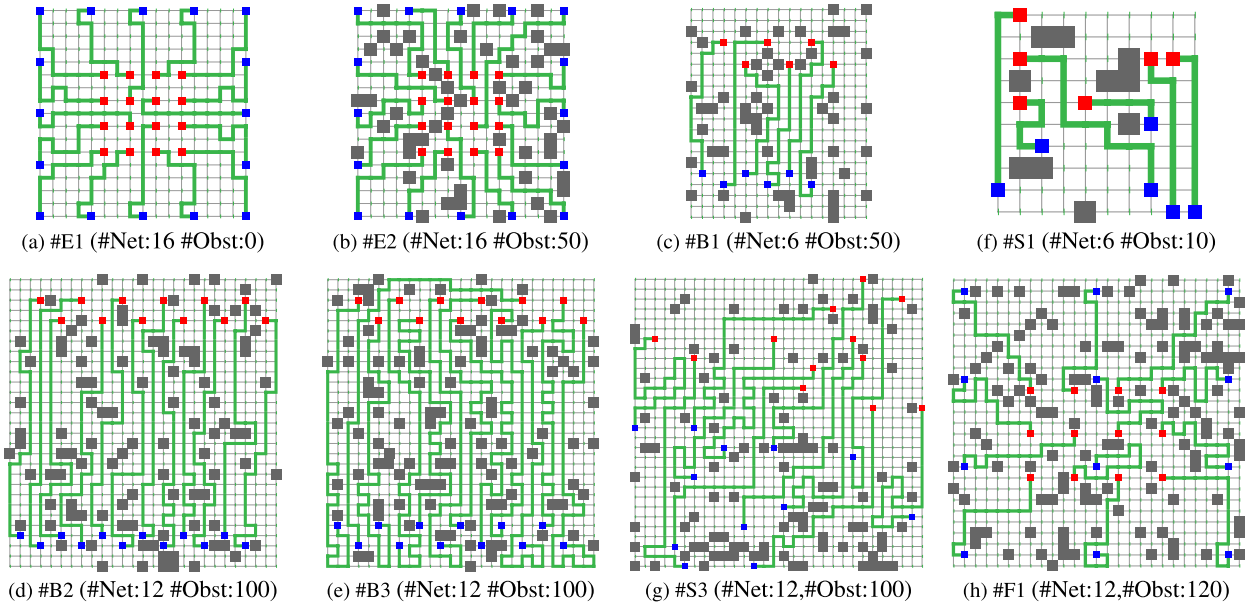
It is confirmed that the length matching is improved from routing patterns corresponding to minimum cost maximum flows. Stage 2 consumes relatively long time in the current implementation. It would be reduced if the number of combinations of vertices checked is reduced by selecting the combinations properly.

6. Conclusion

In this paper, an algorithm that reduces the length difference while keeping the total wire length small is proposed

Table 1 Experimental result.

Instance	Grid	#Net	#Obst	Stage 1 (Tot. Reduction)				Stage 2 (Max. Reduction)				Stage 3 (Min. Increase)			
				L_{tot}	L_{max}	d_{max}	t_1 (sec.)	L_{tot}	L_{max}	d_{max}	t_2 (sec.)	L_{tot}	L_{max}	d_{max}	t_3 (sec.)
#E1	19x19	16	0	136	10	4	0.011	136	10	2	3.414	160	10	0	0.0002
#E2	19x19	16	50	146	18	12	0.004	146	16	10	0.494	154	16	8	0.0001
#B1	19x22	6	50	93	19	9	0.005	93	17	4	0.197	97	17	2	0.0000
#B2	29x31	12	100	338	35	12	0.020	338	30	5	18.049	354	30	1	0.0001
#B3	29x31	12	100	356	46	24	0.040	356	46	20	11.892	420	46	12	0.0004
#S1	10x10	6	10	44	17	14	0.001	44	12	9	0.023	46	12	8	0.0000
#S2	20x20	8	60	91	22	17	0.006	91	16	10	0.557	117	16	2	0.0001
#S3	30x30	12	100	271	51	44	0.053	271	30	19	33.465	355	30	1	0.0004
#F1	29x29	12	120	132	23	20	0.019	132	19	16	7.950	196	19	4	0.0003
Avg.				160.7	24.1	15.6	0.016	160.7	19.6	9.5	7.604	189.9	19.6	3.8	0.0002



(a) #E1 (#Net:16 #Obst:0) (b) #E2 (#Net:16 #Obst:50) (c) #B1 (#Net:6 #Obst:50) (f) #S1 (#Net:6 #Obst:10)

(d) #B2 (#Net:12 #Obst:100) (e) #B3 (#Net:12 #Obst:100) (g) #S3 (#Net:12,#Obst:100) (h) #F1 (#Net:12,#Obst:120)

Fig. 10 Routing patterns.

for a set-pair routing problem. Experiments show the effectiveness of the proposed algorithm. In order to adapt high-speed designs, the post-processing that further improves the obtained routing patterns such as meander wiring, corner rounding, and delay consideration would be required. Our algorithm will lead such the post-processing to meaningful.

Acknowledgments

The authors would like to thank Mr. Yusaku Yamamoto for his collaborative research with the authors when he was at Osaka University and Tokyo Institute of Technology.

References

[1] Y.-K. Ho and Y.-W. Chang, "Multiple chip planning for chip-interposer codesign," Proc. 50th Annual Design Automation Conference, DAC'13, pp.1–6, 2013.
[2] W.-H. Liu, M.-S. Chang, and T.-C. Wang, "Floorplanning and signal assignment for silicon interposer-based 3D ICs," Proc. The 51st Annual Design Automation Conference on Design Automation Conference, DAC'14, pp.1–6, 2014.
[3] M.M. Ozdal and M.D.F. Wong, "Algorithmic study of single-layer bus routing for high-speed boards," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.25, no.3, pp.490–503, 2006.
[4] M.M. Ozdal and M.D.F. Wong, "A length-matching routing algorithm for high-performance printed circuit boards," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol.25, no.12, pp.2784–2794, 2006.
[5] Y. Kohira and A. Takahashi, "CAFE router: A fast connectivity aware multiple nets routing algorithm for routing grid with obstacles," IEICE Trans. Fundamentals, vol.E93-A, no.12, pp.2380–2388, 2010.
[6] T.C. Hu, Integer Programming and Network Flows, Addison-Wesley, Reading, Mass., 1969.
[7] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, North-Holland, Amsterdam, 1976.
[8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, MIT Press and McGraw-Hill, 1990.
[9] M.R. Garey and D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, Freeman and Co., New York, 1979.
[10] E.W. Dijkstra, "A note on two problems in connexion with graphs," Numer. Math., vol.1, no.1, pp.269–271, 1959.
[11] M.R. Kramer and J.v. Leeuwen, "Systolische berechnungen und VLSI," Informatik Spektrum, vol.7, no.3, pp.154–165, 1984.
[12] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," SIAM J. Comput., vol.5, no.4, pp.691–703, 1976.
[13] Y. Shiloach and Y. Perl, "Finding two disjoint paths between two pairs of vertices in a Graph," J. ACM, vol.25, no.1, pp.1–9, 1978.
[14] A. Itai, Y. Perl, and Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," Networks, vol.12, no.3, pp.277–286, 1982.
[15] A. Takahashi, "Practical fast clock-schedule design algorithms," IEICE Trans. Fundamentals, vol.E89-A, no.4, pp.1005–1011, 2006.
[16] R. Bellman, "On a routing problem," Quarterly of Applied Mathematics, vol.16, pp.87–90, 1958.
[17] Y. Kohira, S. Suehiro, and A. Takahashi, "A fast longer path algorithm for routing grid with obstacles using biconnectivity based length upper bound," IEICE Trans. Fundamentals, vol.E92-A, no.12, pp.2971–2978, 2009.

Yuta Nakatani received the B.E. degree from the Tokyo Institute of Technology, Tokyo, Japan, in 2013, where he is currently pursuing the master's degree from the Department of Communications and Computer Engineering, Graduate School of Science and Engineering. His current research interests include VLSI layout design and combinational algorithms.

Atsushi Takahashi received the B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He was at the Tokyo Institute of Technology as a Research Associate from 1991 to 1997, and as an Associate Professor from 1997 to 2009 and from 2012 to 2015. From 2009 to 2012, he was at Osaka University, Suita, Japan, as an Associate Professor. He visited University of California, Los Angeles, U.S.A., as a Visiting Scholar from 2002 to 2003. He is currently a Professor with Department of Communications and Computer Engineering, Graduate School of Science and Engineering, Tokyo Institute of Technology. His current research interests include VLSI layout design and combinational algorithms. He is a member of IEEE, ACM, and IPSJ.