論文 / 著書情報
Article / Book Information

Title	CAFE router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles
Authors	Yukihide Kohira, Atsushi Takahashi
Citation	IEICE Trans. Fundamentals, Vol. E93-A, No. 12, pp. 2380-2388
Pub. date	2010, 12
URL	http://search.ieice.org/
Copyright	(c) 2010 Institute of Electronics, Information and Communication Engineers

IEICE TRANS. FUNDAMENTALS, VOL.E93–A, NO.12 DECEMBER 2010

PAPER Special Section on VLSI Design and CAD Algorithms

CAFE Router: A Fast Connectivity Aware Multiple Nets Routing Algorithm for Routing Grid with Obstacles*

Yukihide KOHIRA[†a)] ***and*** **Atsushi TAKAHASHI**[††], ***Members***

SUMMARY Due to the increase of operation frequency in recent LSI systems, signal propagation delays are required to achieve 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). In the routing design of PCB, the controllability of wire length is often focused on since it enables us to control the routing delay. In this paper, we propose CAFE router which obtains routes of multiple nets with target wire lengths for single layer routing grid with obstacles. CAFE router extends the route of each net from a terminal to the other terminal greedily so that the wire length of the net approaches its target wire length. Experiments show that CAFE router obtains the routes of nets with small length error in short time.

key words: *PCB routing, length-matching routing, trunk routing*

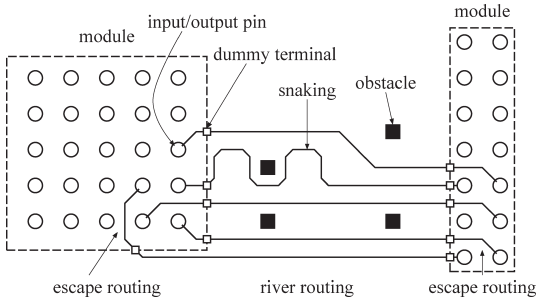


Fig. 1 An example of PCB routing. PCB routing is divided into two parts: river routing and escape routing.

1. Introduction

In recent LSI systems, due to the increase of operation frequency, signal propagation delays are requested to achieve the specifications with very high accuracy [2]–[5]. In order to achieve the severe requirements, signal propagation delay is taken into account in the routing design of PCB (Printed Circuit Board). The signal propagation delay of a net in modules such as LSI and other devices which are put on PCB consists of the routing delay caused by wires and the gate delay caused by gates, and it 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. As the clock frequency increases, the routing method is required to have higher controllability on wire length [2]. The wire length of a net is easily controlled if a large area is reserved for the net unless the wire length of the net is requested to be shorter than the distance between terminals of the 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.

In this paper, we consider the length-matching routing problem for multiple nets in a single layer with obstacles for PCB routing. In this problem, the set of two terminal nets with target wire length is given. The obtained routes of nets must not intersect each other. The length error of a net is the difference between target and actual wire lengths. The objective of this problem is the minimization of the sum of the absolute length errors of nets.

The routing area of PCB has a lot of obstacles since modules are put on. The obstacle density of an area where a module with a lot of pins is put on is high since pins are obstacles. The obstacle density of the rest area is relatively low though there are various kinds of obstacles. The routing area of PCB is often divided into several subareas depending on whether a module with a lot of pins is put on. Routing problems corresponding to subareas where a module with a lot of pins is put on and others are called escape routing and river routing, respectively (Fig. 1).

In escape routing, a net consists of a terminal inside the routing area and a terminal on the boundary of the routing area. The flexibility of the route of a net is limited, and the net is usually requested to just escape the routing area or to minimize the total length. The escape routing methods in [6]–[10] do not take the target wire length of a net into account.

While, in river routing, a net consists of terminals on the boundary of the routing area. The flexibility of the route of a net is high, and the net can make a detour to increase its length without losing the connectivity of other nets. The river routing methods in [11], [12] minimize the total wire length of nets without taking the length of each net into con-

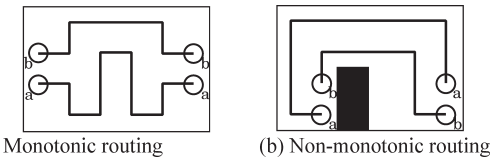
Manuscript received March 17, 2010.
Manuscript revised June 21, 2010.

[†]The author is with the School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu-shi, 965-8580 Japan.
[††]The author is with the Graduate School of Engineering, Osaka University, Suita-shi, 565-0871 Japan.

*The preliminary version was presented at [1].

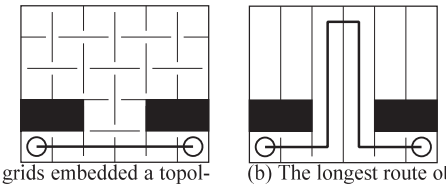
a) E-mail: kohira@u-aizu.ac.jp

DOI: 10.1587/transfun.E93.A.2380



(a) Monotonic routing (b) Non-monotonic routing

Fig. 2 The method in [4] can obtain monotonic routing (a), but it cannot obtain non-monotonic routing (b).



(a) BSG grids embedded a topology (b) The longest route obtained by enlarging BSG grids

Fig. 3 The method in [5] cannot use the whole of routing area (b) according to BSG grids embedded the topology (a).

sideration. The river routing methods in [3]–[5] take the target wire length of each net into consideration. However, they do not work well when the routing area contains obstacles.

The method in [3] divides the routing area and assigns a divided area to each net so that the target wire length of a net is realized within the assigned area. Whether the target length is realized in an area is easily evaluated if the area contains no obstacle. However it is not trivial when the area contains obstacles. No discussion on obstacle is given in [3], and the target length might not be realized in an assigned area if it contains obstacles. The method in [4] realizes the target wire lengths of nets by using a Lagrangian relaxation technique that mediates the routing area among the routes of nets by assuming the route of a net is monotonic. However, it can not be used when the routing area contains obstacles since the non-monotonic route might be required (See Fig. 2). BSG router [5] is a gridless routing method in which the routing area is assigned to each net by using BSG-structure. In this method, BSG grids are assigned to a net according to the given routing topology of the net, and enlarged so that the target length of the net is achieved within the area corresponding to the assigned BSG grids. When the routing area contains no obstacle, the target length of a net is realized almost independent of the embedding of the routing topology onto BSG-structure as the authors claim. However, when the routing area contains obstacles, an achievable length would depend on the embedding of the routing topology onto BSG-structure, and the target length might not be realized. For example, if the routing topology of a net is embedded onto BSG grids as shown in Fig. 3(a), the maximum length is achieved when BSG grids are modified as shown in Fig. 3(b), and the whole area can not be assigned to the net. It seems to be difficult to find a BSG grid assignment that achieves the target lengths when the routing area contains obstacles.

In this paper, the trunk routing problem which is a subproblem of the river routing problem is focused on. Trunk routing problem is one of the key problems since it is derived when nets are defined between two modules. The length-matching routing problem for trunk routing problem is NP-hard since its sub-problem, the longest path problem for a single net in a single layer with obstacles, is NP-hard [13]. Therefore, a fast heuristic algorithm CAFE (Connectivity Aware Frontier Exploration) router is proposed for trunk routing problem. CAFE router guarantees that the connection of every net is realized if a feasible instance is given and takes the target wire length of each net into consideration. In CAFE router, the route of a net is fixed by the movement of the frontier of the net from a terminal of the net to the other terminal of the net one by one so that the wire length of each net approaches its target wire length while keeping the connectivity.

CAFE router cannot handle an instance not in trunk routing as it is. However, CAFE can handle it if it is transformed to a feasible instance in trunk routing. In Sect. 5, wall that restricts the flexibility of the route of a net is introduced to obtain a trunk routing instance from a escape routing instance. Although the generation of wall is out of scope of this paper, the effect of wall is discussed in Sect. 5. Experiments show that CAFE router obtains the routes of nets in short computation time and that the error between the realized wire length and the target wire length of a net is small if reasonable target wire lengths are given. However, CAFE router is a greedy heuristics and has a room for improvement. Experiments also show that the routes of nets whose error are close to zero are obtained by applying R-Flips and S-Flips in post-processing.

2. Preliminaries

2.1 Grid Graph and Length of Route

In this paper, the trunk routing problem which is a subproblem of river routing problem is handled. In our routing problem, a routing area consists of a single layer and contains obstacles. A routing area is divided into unit squares by horizontal lines and vertical lines. The intervals of horizontal lines and vertical lines are set to unit length which corresponds to the minimum distance between two wires. In the following, a routing area is represented by a grid graph. A vertex in a grid graph corresponds to a unit square in a routing area. In a grid graph, two vertices are connected by an edge if two unit squares corresponding to them share their side where a wire can pass. Graph terminologies [14] are used in explanation, but a routing grid representation is used in figure for simplicity and readability as shown in Fig. 4. 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*. Note that a routing area with an arbitrary shape can be represented by setting obstacles on the boundaries of the routing area.

The set of nets N is given as input. Each net consists of two terminals put on the boundary of the routing area in

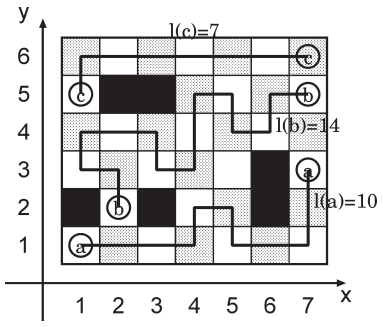


Fig. 4 An example of river routing problem and river routes.

river routing problem. A terminal is on on-square and drawn by a circle in figures.

Each square of a routing grid is referred by its coordinate. The coordinate of the leftmost-bottom square is defined as $(1, 1)$. For example, terminals of net a are $(1, 1)$ and $(7, 3)$ in Fig. 4.

A route is represented by horizontal and vertical line segments that pass on-squares. The *length* $l(n)$ of a net $n \in N$ is defined by the number of edges of the route in the grid graph. For example, lengths of net a, b, and c shown in Fig. 4 are $l(a) = 10$, $l(b) = 14$, and $l(c) = 7$, respectively.

The problem we concern is to find routes of multiple nets which minimize the error between the length $l(n)$ and the *target length* $l_{tar}(n)$ which is given for each net n. Since the routing area is bipartite, the parity of the length of any route of a net is the same. Note that when 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, any route of a net passes white and gray squares alternatively. The parity of $l(n)$ depends on the combination of the location of two terminals. If the parity of $l(n)$ is different from the parity of $l_{tar}(n)$, then the difference between $l(n)$ and $l_{tar}(n)$ is at least one. In order to evaluate the difference between $l(n)$ and $l_{tar}(n)$ appropriately, we define the *error* $l_{err}(n)$ for each net n so that the inevitable difference is not taken into account. If the parities of $l(n)$ and $l_{tar}(n)$ are the same, $l_{err}(n) = l(n) - l_{tar}(n)$. Otherwise, $l_{err}(n) = l(n) - l_{tar}(n) - 1$ when $l(n) > l_{tar}(n)$ and $l_{err}(n) = l(n) - l_{tar}(n) + 1$ when $l(n) < l_{tar}(n)$. In addition, the *average error* $L^{ave}_{err}(N)$ and the *worst error* $L^{worst}_{err}(N)$ for N are defined as follows:

$$L^{ave}_{err}(N) = \frac{\sum_{n \in N} |l_{err}(n)|}{|N|}.$$

$$L^{worst}_{err}(N) = \begin{cases} l^P_{err}(N) & (l^P_{err}(N) \geq l^M_{err}(N)) \\ -l^M_{err}(N) & (l^P_{err}(N) < l^M_{err}(N)), \end{cases}$$

where,

$$l^P_{err}(N) = \max_{n \in N} l_{err}(n)$$

$$l^M_{err}(N) = \max_{n \in N}(-l_{err}(n)).$$

For example, in Fig. 4, assume that $l_{tar}(a) = l_{tar}(b) = l_{tar}(c) = 11$. Then, $l_{err}(a) = 0$, $l_{err}(b) = 2$, $l_{err}(c) = -4$, $L^{ave}_{err}(N) = 2$, and $L^{worst}_{err}(N) = -4$.

2.2 Feasibility of River Routing Problem

A route of a net is said to be *feasible* if it passes each on-square at most once, does not pass any off-squares, and connects two terminals of the net without intersecting routes of the other nets. A set of routes is said to be *feasible* if it consist of feasible routes of all nets. Note that a set of feasible routes for a set of nets N corresponds to *vertex-disjoint paths*, where the number of paths is $|N|$, in the grid graph. An instance of river routing problem is said to be *feasible* if it has a set of feasible routes.

Let S and T be the sets of the terminals of the set of nets N such that each net has a terminal in S and another terminal in T. If the maximum number of vertex-disjoint paths each of which connects a vertex in S and a vertex in T is less than the number of nets, then the instance is infeasible. Even if the maximum number is equal to the number of nets, it does not mean that the instance is feasible since they may connect terminals of different nets. However, it enables us to reject several infeasible instances. Furthermore, when an instance satisfies a certain condition, the fact that the maximum number is equal to the number of nets means that the instance is feasible.

Given two disjoint vertex sets in a graph, the problem finding the maximum number of vertex-disjoint paths each of which connects a vertex in a vertex set and a vertex in another vertex set is called the disjoint paths problem and it is known to be NP-hard even if the graph is planar [15]. However, it is not clear whether the disjoint paths problem in grid graph is NP-hard.

2.3 Definition of Trunk Routing Problem

In this paper, we handle the trunk routing problem which is sub-problem of river routing problem and is one of the key problems since it is derived when nets are defined between two modules. A river routing problem that satisfies the *trunk routing topology condition* defined below is called a trunk routing problem. The length-matching problems in river routing and that in trunk routing are NP-hard as mentioned Sect. 1. Although it is not clear whether the disjoint paths problem in grid graph is NP-hard, the disjoint paths problem in trunk routing is P. Here, trunk routing topology condition and the definition of trunk routing problem which we concern are described.

In a river routing problem, the terminals are on the boundary of the routing region. Let the boundary terminal sequence be the sequence of terminals obtained along the boundary. The length of the boundary terminal sequence is $2n$, where n is the number of nets. The trunk routing topology condition is defined in the following.

Trunk routing topology condition

- All terminals are put on the boundary of the routing area.
- The boundary terminal sequence can be divided into the source terminal sequence S and the sink terminal sequence T, where T is the reverse of S and vice versa.

For an instance that satisfies the trunk routing topology condition, the source terminal sequence S and the sink terminal sequence T are defined. The terminals in S are called *source terminals* and those in T are called *sink terminals*. For example, in Fig. 2, sources are the terminals on squares $(1, 1)$, $(2, 2)$, and $(1, 5)$. Similarly, sinks are the terminals on squares $(7, 3)$, $(7, 5)$, and $(7, 6)$.

Trunk routing problem which we concern is defined as follows.

Trunk routing problem

input target length for each net, grid graph, source terminals, and sink terminals, where the instance satisfies the trunk routing topology condition
output a set of feasible routes
objective minimize average error or worst error

2.4 Complexity of Feasibility Check in Trunk Routing

Even if an instance satisfies the trunk routing topology condition, a set of feasible route is not always obtained. However, if an instance satisfies the trunk routing topology condition, the maximum number of vertex-disjoint paths is equal to the amount of maximum flow in the corresponding *flow graph*. Therefore, in trunk routing, feasibility can be checked in the polynomial time, since the amount of maximum flow can be calculated in the polynomial time.

The flow graph with vertex capacities is defined as follows. The vertex set corresponds to on-squares in the routing grid. The capacity of each vertex is set to one. The edge set corresponds to edges in the routing grid. If two grids are adjacent in the grid graph, two vertices are connected by two directed edges in opposite directions. In addition, the primary source and the primary sink are added. The primary source is connected to each source terminal and the primary sink is connected from each sink terminal. For example, the flow graph corresponding to the shown in Fig. 4 is shown in Fig. 5.

If an instance satisfies the trunk routing topology condition, each unit flow in a maximum flow corresponds to a feasible route of a net when the amount of a maximum flow in this flow graph is equal to the number of nets in the instance. Note that the flow graph is planar and that a unit flow does not intersect each other since the capacity of a vertex is one. On the other hand, when the amount of a maximum flow in this flow graph is less than the number of nets in the instance, no feasible route of a net exists.

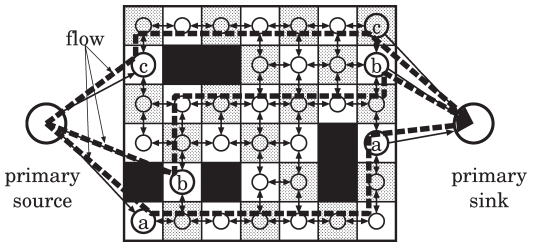


Fig. 5 Flow graph corresponding to the instance shown in Fig. 4 and maximum flow.

3. CAFE Router

In this section, we propose a fast heuristic algorithm CAFE (Connectivity Aware Frontier Exploration) router to obtain a feasible routes whose errors between the lengths of obtained routes and target lengths are small. In CAFE router, the route of a net is fixed by the movement of the *frontier* of the net from the source terminal to the sink terminal of the net. The outline of CAFE router is shown in Fig. 6.

While CAFE router fixes the route of a net, the route of a net is divided into two parts, the fixed part and the unfixed part. The route from the source terminal to the on-square just before the frontier of a net is the fixed part. An on-square which is passed by the fixed part of a net is called *passed*. An on-square which has the frontier of a net is called *occupied*. An on-square which is neither passed nor occupied is called *free*.

At first, CAFE router chooses a *focused* net whose frontier is moved. In our implementation in experiments, a net with the maximum remained length is chosen as the focused net, where the remained length is the difference between the target length and the length of the fixed part of the net. For example, an instance shown in Fig. 4 is shown in Fig. 7(a), where $l_{tar}(a) = 8, l_{tar}(b) = 10, l_{tar}(c) = 7$. The frontier of each net represented by triangle in Fig. 7 is set to the source terminal of the net. Net b is chosen as the focused net in our experiment.

Next, CAFE router checks whether the connectivity is kept in each partial solution obtained by moving the frontier to an adjacent free on-square. The instance that corresponds to a partial solution obtained after the movement of the frontier of the focused net satisfies the trunk routing topology condition. The connectivity of each partial solution is checked by using the flow graph modified as follows: passed on-squares are regarded as off-squares, and occupied squares are set to source terminals. If the amount of the maximum flow is equal to the number of nets, the connectivity is kept in the partial solution, and the adjacent free square is called *feasible* in terms of the movement. Otherwise, the connectivity is lost, and the adjacent free square is called *infeasible* in terms of the movement.

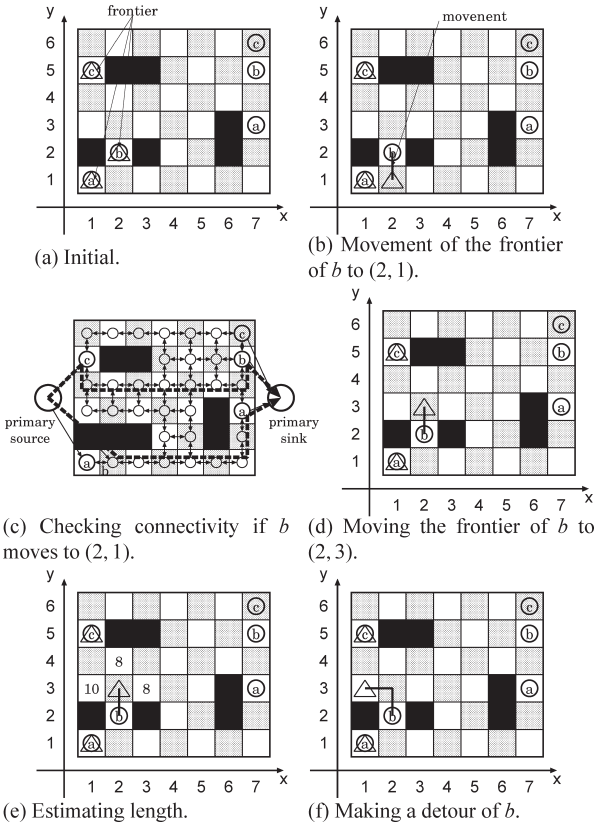
After the procedures, at least one feasible adjacent free on-square of the focused net is remained since the instance before the movement of the frontier of the focused net holds

Input : target length for each net, grid graph, source terminals, and sink terminals, where the instance satisfies the trunk routing topology condition
Output : a set of feasible routes
Objective : minimize average error or worst error
Step 1 : The frontier of each net is set to the source terminal of the net.
Step 2 : Until each frontier arrives at the sink terminal, repeat the following.

1. Choose a net with maximum remained length as the focused net.
2. Select the set of feasible on-squares from adjacent free on-squares of the frontier of the focused net by checking the connectivity.
3. Choose a feasible free adjacent on-square such that the estimated error is minimum among the set of feasible on-squares.
4. Move the frontier of the focused net to the selected on-square.

Step 3 : Return the movement of frontiers.

Fig. 6 Outline of CAFE router.



(a) Initial.
(b) Movement of the frontier of b to $(2, 1)$.
(c) Checking connectivity if b moves to $(2, 1)$.
(d) Moving the frontier of b to $(2, 3)$.
(e) Estimating length.
(f) Making a detour of b.

Fig. 7 An example of applying CAFE router to the instance shown in Fig. 4, where $l_{tar}(a) = 8, l_{tar}(b) = 10, l_{tar}(c) = 7$.

the connectivity and the feasible route set exists. For example, suppose that the frontier of net b moves from $(2, 2)$ to $(2, 1)$ (Fig. 7(b)). A maximum flow in the corresponding flow graph are shown in Fig. 7(c). Since the maximum flow is less than the number of nets, connectivity is lost if the frontier of net b moves to $(2, 1)$. In CAFE router, the frontier of net b does not move to $(2, 1)$, but moves to $(2, 3)$ (Fig. 7(d)).

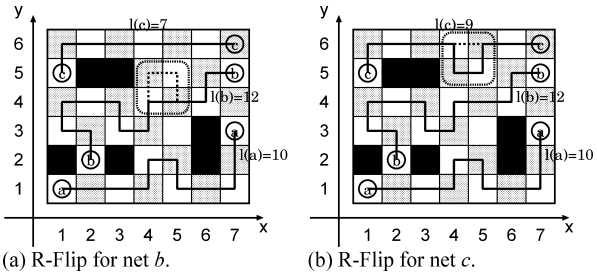
Next, CAFE router selects a square among feasible on-squares to which the frontier is moved. The estimated length of a net is changed when the frontier moves to a feasible on-square. CAFE router selects a square so that the difference of the estimated length and the target length is minimum. The length of the focused net is estimated as the sum of the length of the fixed part of the focused net and the estimated length of the unfixed part of the focused net. The length of the unfixed part is estimated by the shortest path length from the square on the frontier to the sink terminal without considering the unfixed parts of the other nets in free squares. When the remained length is larger than the estimated length of the unfixed part of the focused net, CAFE router chooses a feasible adjacent free on-square with maximum shortest path length since the route which is connected by a shortest path becomes smaller than the target length of the focused net. On the other hand, when the remained length is less than the estimated length of the unfixed part of the focused net, CAFE router chooses a feasible adjacent free on-square with minimum shortest path length since the route which is connected by a shortest path becomes larger than the target length of the focused net. For example, after the frontier of net b moves from $(2, 2)$ to $(2, 3)$ (Fig. 7(d)), net b is chosen as the focused net again since the remained length of net b is 9 and it is maximum. The frontier of net b can move from $(2, 3)$ to all adjacent on-squares $(1, 3)$, $(2, 4)$ and $(3, 3)$ since the connectivity is kept. The estimated length of each adjacent on-square is shown in Fig. 7(e). The difference of the estimated length and the target length is 0 so the frontier of net b moves from $(2, 3)$ to $(1, 3)$ (Fig. 7(f)). These procedures are repeated until the frontier of each net arrives at the sink terminal of the net.

Here, we discuss the time complexity of CAFE router. Let n and m be the number of on-squares in the grid graph and the number of nets, respectively. The frontiers of all nets arrive at sink squares by at most n movements in total. In each movement, the number of computation of checking connectivity is at most four. Therefore, since the time complexity of checking connectivity is $O(mn)$ the time complexity of CAFE router is $O(mn^2)$.

4. R-Flip and S-Flip

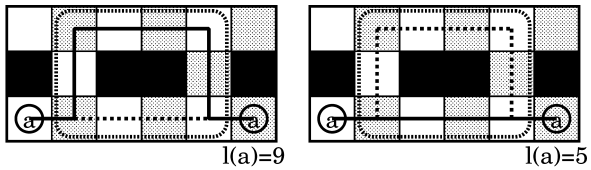
CAFE router obtains a feasible route set such that its errors between the lengths of obtained route and the target lengths are small. However, since CAFE router is heuristic, the error of a route obtained by CAFE router may be reduced by post-processing. Here, R-Flip [13] and S-Flip that modify the route of a net locally are introduced which would be used in post-processing to reduce the error of a route.

R-Flip [13] is a kind of Flip introduced in [16]. A R-Flip modifies a partial route of a net within two times two unit-squares where no route of other nets passes if the partial route passes either one or three of four sides shared by



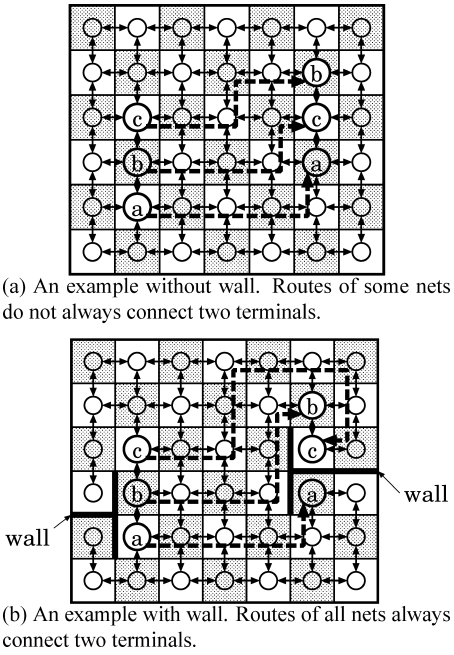
(a) R-Flip for net *b*. (b) R-Flip for net *c*.

Fig. 8 Applying R-Flip for the routes shown in Fig. 4.



(a) Before S-Flip. (b) After S-Flip.

Fig. 9 Applying S-Flip.



(a) An example without wall. Routes of some nets do not always connect two terminals.

(b) An example with wall. Routes of all nets always connect two terminals.

Fig. 10 An instance with terminals in routing area. The primary source and the primary sink are omitted for simplicity.

these unit-squares. By applying a R-Flip, the route comes not to pass a side shared by these unit-squares that the route passed, and pass a side that the route did not pass. The length of route of a net either increases by two or decreases by two. When the length of route of a net is larger than the target length, the absolute error is decreased if R-Flip is applied to the net so that its length is decreased (Fig. 8(a)). When the length of route of a net is smaller than the target length, the absolute error is decreased if R-Flip is applied to the net so that its length is increased (Fig. 8(b)).

S-Flip is an enhancement of R-Flip. In a S-Flip, a straight segment of route of a net is shifted (Fig. 9). If the length of a straight segment shifted by a S-Flip is one, a route obtained by the S-Flip is obtained by applying R-Flip several times. In our implementation, S-Flip is used to reduce the length of route of a net, and applied if no design rule violation occurs and if the error of the net remains non-negative. R-Flip and S-Flip are greedily applied to nets whose errors are not 0 so that the absolute error of a net is reduced until no absolute error is reduced. A net is chosen arbitrary, and the candidate of R-Flips and S-Flips of the net are searched along the route of the net from its source terminal to its sink terminal.

5. Wall

CAFE router obtains a feasible route set if an instance that satisfies the trunk routing topology condition is feasible. However, even if a feasible route set exists for an instance, CAFE router can not be applied to the instance if it does not satisfy the trunk routing topology condition. In order to utilize CAFE router to obtain a feasible routing set for such instances, a modification of an instance to an instance that satisfies the trunk routing topology condition is discussed. Here, we introduce a *wall* that prohibits a route from passing the side of unit-square where the wall is inserted. If walls are inserted to the routing area so that the derived instance satisfies the trunk routing topology condition, CAFE router can be applied to the derived instance. For example, the instance shown in Fig. 10(a) which does not satisfy the trunk routing topology condition becomes to satisfy the trunk routing topology condition by inserting walls as shown in Fig. 10(b). In the grid graph, edges that correspond to the side of unit-square where a wall is inserted are removed.

Even if an instance is modified to satisfy the trunk routing topology condition, the derived instance may become infeasible. Wall insertion that keeps the feasibility and that modifies the instance to satisfy the trunk routing topology condition is not easy. Wall insertion remains in our future works.

6. Experimental Results

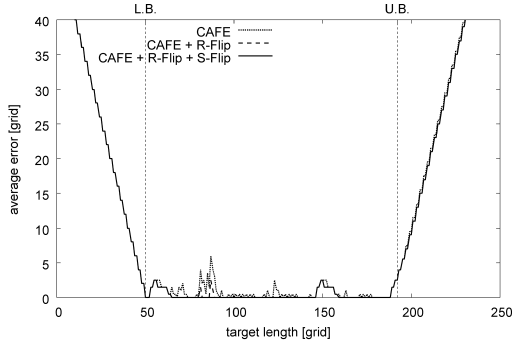
CAFE router, R-Flip, and S-Flip are implemented in C++, compiled by gcc4.2.4, and executed on a PC with 2.93 GHz Intel Core i7 CPU and 6 GB RAM. They are applied to three sample data generated by referring to industrial designs. Among three sample data, data 1 satisfies the trunk routing topology condition, but data2 and data3 do not satisfy the trunk routing topology condition since terminals are not put on the boundary of the routing area. In order to derive instances that satisfy the trunk routing topology condition from data2 and data3, walls were inserted to data2 and data3 according to the escape routing generated artificially.

In experiments, the target lengths of all the nets are set to the same. If the target length is set to either too small or too large, it is impossible to achieve the target length. If the target length of a net is less than the distance between termi-

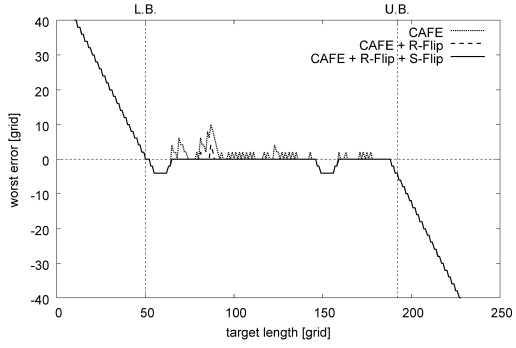
Table 1 Data of samples.

	#nets	area	#on	L.B.	U.B.
data1	4	28x28	771	50	192
data2	9	13x53	655	35	72
data3	13	61x130	4489	165	350

#on the number of on-squares
L.B. a lower bound of target length such that $L_{err}^{worst}(N) = 0$ (the maximum distance between terminals among nets)
U.B. an upper bound of target length such that $L_{err}^{worst}(N) = 0$ (#on-squares divided by #net)



(a) Average error.

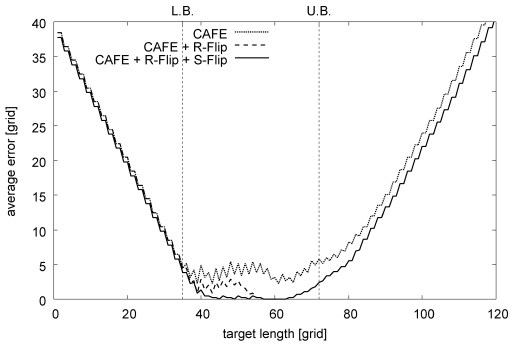


(b) Worst error.

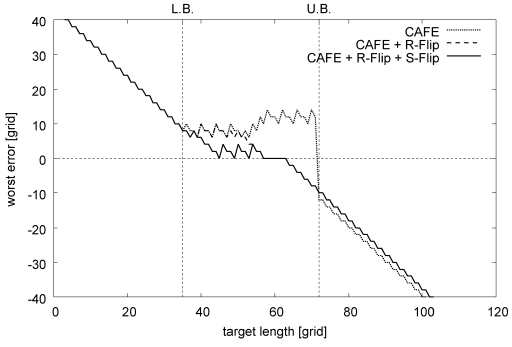
Fig. 11 Relation between target length and error for data1.

nals of the net, it is apparent that the target length can not be achieved. Also, if the target length of a net is larger than the number of on-squares, the target length can not be achieved. The target length of a net is said to be significant if it is not apparent whether the target length is achievable or not. In the following, we regard the target length as significant if it is between the lower bound and the upper bound defined below. The lower bound is the maximum distance between terminals among nets. The upper bound is the number of on-squares divided by the number of nets. In experiments, the target lengths set to from small enough compared to the lower bound to large enough compared to the upper bound. These data of samples are shown in Table 1.

The relations between the target lengths and average errors and those between the target lengths and worst errors in data1, data2, and data3 are shown in Fig. 11, Fig. 12, and Fig. 13, respectively. If the target lengths are set to small enough, both the average errors and the worst errors are large positive, since the target length of each net is shorter

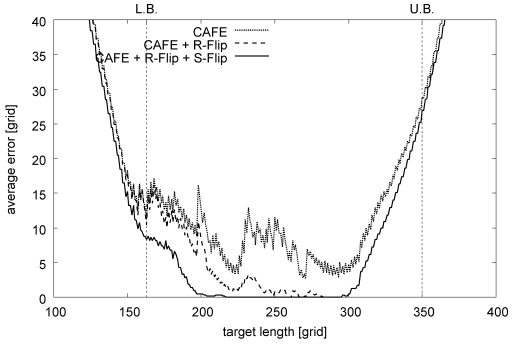


(a) Average error.

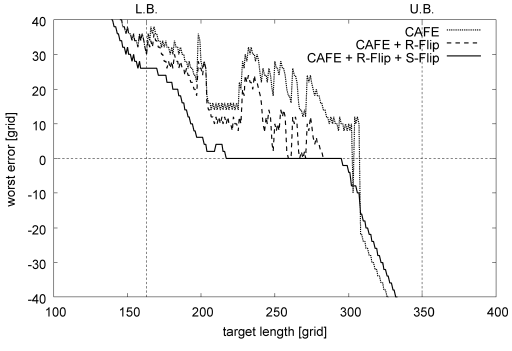


(b) Worst error.

Fig. 12 Relation between target length and error for data2.



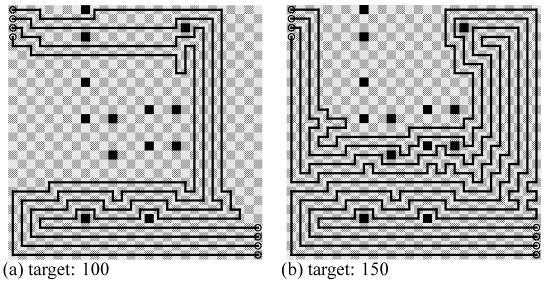
(a) Average error.



(b) Worst error.

Fig. 13 Relation between target length and error for data3.

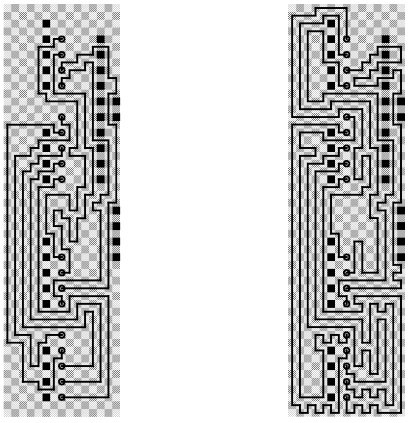
than the shortest path length of the net and it cannot be realized. On the other hand, if the target lengths are set to large enough, the average errors are large positive and the



(a) target: 100
average error: 0
worst error: 0
comp. time: 0.36[s]

(b) target: 150
average error: 2.5
worst error: −4
comp. time: 0.41[s]

Fig. 14 Results for data1 by CAFE+R-Flip+S-Flip.



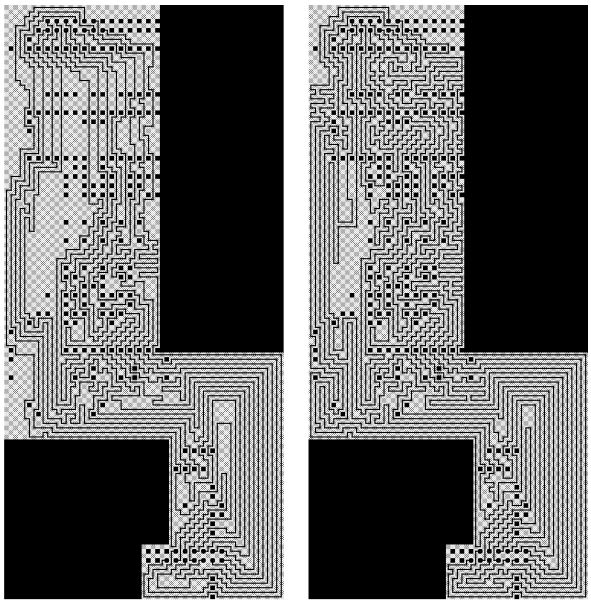
(a) target: 50
average error: 0.44
worst error: 4
comp. time: 0.38[s]

(b) target: 70
average error: 1.56
worst error: −8
comp. time: 0.45[s]

Fig. 15 Results for data2 by CAFE+R-Flip+S-Flip.

worst errors are small negative, since the sum of the target lengths of all nets is larger than the routing area and the target length of each net cannot be realized. The closer both the average errors and the worst errors are to 0, the better the effect of routing method is. In figures, the errors of the routes obtained by CAFE router only are represented by CAFE, those obtained by R-Flip after applying CAFE router are represented by CAFE+R-Flip, and those obtained by S-Flip after applying CAFE router and R-Flip are represented by CAFE+R-Flip+S-Flip. Note that R-Flip improves both negative and positive errors, but S-Flip improves only positive errors.

In data1, the average errors and the worst errors by CAFE router are small. The results show that CAFE router is very effective in the routing area except the module area in PCB routing problem. In data2 and data3, the worst errors by CAFE router tend to be positive. It means that the lengths obtained routes are larger than the target lengths. It is caused why the estimation of length by shortest path length is wrong. When the frontier of each net is close to the source terminal, the route takes many detours since the target length is larger than the shortest path length. On the other hand, when the frontier of each net is close to the



(a) target: 250
average error: 0
worst error: 0
comp. time: 23.89[s]

(b) target: 300
average error: 0.46
worst error: −4
comp. time: 24.45[s]

Fig. 16 Results for data3 by CAFE+R-Flip+S-Flip.

sink terminal, the route cannot take detours since the target length is close to the shortest path length. However, since the route cannot be connected by the shortest path and must take detours in the module area, the obtained routes are larger than the target length. R-Flip and S-Flip after applying CAFE router is much better than CAFE router since R-Flip and S-Flip can reduce the lengths of routes. The obtained routes are shown in Fig. 14, Fig. 15, and Fig. 16, respectively. The comp. time means the computation time of CAFE router in a caption of each figure. The computation time of R-Flip and S-Flip is less than 0.01[s] in each instance. The average errors are much less than the absolute worst errors. It means that the errors of nets except the net with the worst error close to 0.

7. Conclusion

In this paper, we propose a fast heuristic algorithm CAFE router in order to obtain the route with small errors between lengths of routes and target lengths. In CAFE router, a route of each net is fixed by movement of the frontier from a terminal to the other terminal of each net, holding the connectivity of all nets in the routing area. The frontier of a net with maximum remained length moves from the current square to an adjacent square iteratively so that the estimated wire length of the net approaches the target wire length of the net and so that the connectivity of nets is kept. Experiments show that CAFE router obtains the routes of nets in short computation time and errors between lengths of the routes obtained by CAFE router and target lengths of nets are small.

CAFE router restricts the instance according to the arrangement of terminals and it has large errors in some instances. The restriction relaxation according to the arrangement including the wall generation method and improvements of CAFE router according to the errors will be in our future works.

Acknowledgements

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

References

[1] Y. Kohira and A. Takahashi, "CAFE router: A fast connectivity aware multiple nets routing algorithm for routing grid with obstacles," ASPDAC, pp.281–286, 2010.
[2] L.W. Ritchey, "Busses: What are they and how do they work?," Printed Circuit Design Magazine, Dec. 2000.
[3] M.M. Ozdal and M.D.F. Wong, "Algorithmic study of single-layer bus routing for high-speed boards," IEEE Trans. Comput.-Aided Des. Integrated 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 Des. Integrated Circuits Syst., vol.25, no.12, pp.2784–2794, 2006.
[5] T. Yan and M.D.F. Wong, "BSG-route: A length-constrained routing scheme for general planar topology," IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst., vol.28, no.11, pp.1679–1690, 2009.
[6] Y. Kajitani, "The potential router," IEICE Technical Report, VLD2006–129, 2007.
[7] M. Inagi, Y. Takashima, and Y. Kajitani, "Escape fitting between a pair of pin-sets," IEICE Technical Report, VLD2006–130, 2007.
[8] T. Yan and M.D.F. Wong, "Untangling twisted nets for bus routing," ICCAD, pp.396–400, 2007.
[9] L. Luo and M.D.F. Wong, "Ordered escape routing based on boolean satisfiability," ASP-DAC, pp.244–249, 2008.
[10] L. Luo and M.D.F. Wong, "On using SAT to ordered escape problems," ASP-DAC, pp.594–599, 2009.
[11] R.Y. Pinter, "On routing two-point nets across a channel," DAC, pp.894–902, 1982.
[12] C.P. Hsu, "Channel river routing algorithm," DAC, pp.578–583, 1983.
[13] 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, Dec. 2009.
[14] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications, North-Holland, 1976.
[15] M.R. Garey and D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, W.H. Freeman & Company, 1979.
[16] Y. Kubo, Y. Takashima, S. Nakatake, and Y. Kajitani, "Self-reforming steiner tree by flip and applications to VLSI interconnection," IPSJ J., vol.41, no.4, pp.881–888, 2000.

Yukihide Kohira received his B.E., M.E., and D.E. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2003, 2005, and 2007, respectively. He had been a researcher of Department of Communications and Integrated Systems in Tokyo Institute of Technology from 2007 to 2009. He is currently with the School of Computer Science and Engineering, the University of Aizu, as an assistant professor since 2009. His research interests are in VLSI design automation and combinational algorithms. He is a member of IEEE and IPSJ.

Atsushi Takahashi received his 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 had been with the Tokyo Institute of Technology as a research associate from 1991 to 1997 and had been an associate professor from 1997 to 2009. He visited University of California, Los Angeles, U.S.A., as a visiting scholar from 2001 to 2002. He is currently with Division of Electrical, Electronic and Information Engineering, Graduate School of Engineering, Osaka University, as an associate professor since 2009. His research interests are in VLSI layout design and combinational algorithms. He is a member of IEEE and IPSJ.