An Improved Optimal Algorithm for Bubble-Sorting-Based Non-Manhattan Channel Routing

Jin-Tai Yan

***Abstract*— It is well known that a non-Manhattan channel router always uses fewer routing tracks than a Manhattan router in a channel. To our knowledge, for a bubble-sorting-based non-Manhattan channel routing (BSNMCR) problem, Chaudhary's $O(kn^2)$ heuristic algorithm [8] and Chen's $O(k^2n)$ optimal algorithm [9] have been, respectively, proposed, where n is the number of terminals and k is the number of routing tracks in a channel. However, the time complexity of the two algorithms is in $O(n^3)$ time in the worst case. In this paper, based on optimality-oriented swap-direction selection in an optimal bubble-sorting solution, an improved optimal algorithm for a BSNMCR problem is proposed, and the time complexity of the proposed algorithm is proven to be in $O(kn)$ time and in $O(n^2)$ time in the worst case.**

***Index Terms*— Bubble sorting, channel routing, optimal algorithm, physical design.**

I. Introduction

IT IS WELL known that channel routing (CR) plays an important role in very large scale integration (VLSI) design automation, and the CR problem has been extensively studied. Generally speaking, most of the routers only use horizontal and vertical wires to complete the connection of all the routing nets in a channel, i.e., most of the channels are routed in a Manhattan routing model [1]–[5]. As VLSI technology advances, the fabrication process does not preclude a layout style in a non-Manhattan routing model. In fact, a non-Manhattan channel router always uses fewer routing tracks than a Manhattan router in a channel. Hence, new non-Manhattan CR problems have been formulated [6], [8], and nonoptimum and optimal non-Manhattan algorithms [6]–[9] have been proposed.

For a bubble-sorting-based non-Manhattan channel, the basic concept in Wang's algorithm [7] is to interchange a pair of adjacent nets using two wires, one in the $+45°$ direction and the other in the $-45°$ direction if two adjacent nets are in a wrong order. On the other hand, the nets propagate to the next track over vertical wires if two adjacent nets are in a right order. Hence, the routing process in a channel is determined by a sequence of passes of interchanging nets. In each pass, a net is only moved one position to the left, one position to the right, or remains at the same position. Basically, each pass of interchanging nets can be implemented by one routing track in a non-Manhattan channel. As a result, a bubble-sorting-based non-Manhattan channel will be routed by mapping each pass onto one routing track.

To minimize the number of routing tracks in a bubble-sorting-based non-Manhattan channel, Chaudhary's algorithm [8] further releases the constraint of moving at most one position to the left or the right during one pass. Hence, the operation of swapping nets can be propagated in one pass, and all the nets can be moved over longer distances during one pass. Basically, the swap-direction of any pass in Chaudhary's algorithm [8] depends on the number of nonzero elements in the right and left inversion tables. If the number of nonzero elements in the left (right) inversion table is greater than that in the right (left) inversion table, one right (left) swap pass will be performed from left to right (right to left). On the other hand, if the number of nonzero elements in the left inversion table is the same as that in the right inversion table, one left swap pass will be performed from right to left. Hence, Chaudhary's algorithm takes $O(kn^2)$ time to route a bubble-sorting-based non-Manhattan channel, where k is the number of routing tracks and n is the number of terminals in a channel. In Fig. 1, a Manhattan routing result and non-Manhattan routing results by Wang's algorithm [7] and Chaudhary's algorithm [8] for a channel are illustrated, respectively.

Recently, Chen *et al.* [9] proposed an optimal algorithm to solve a bubble-sorting-based non-Manhattan channel routing (BSNMCR) problem. Chen *et al.* prove a theorem, $LRV = RLV$, to explain no sorting-order in an optimal bubble-sorting solution (BSS), and to further generate a binary decision diagram for an optimal BSS. If a left-swap pass and a right-swap pass are applied to a bubble-sorting problem at the same time, Chen's algorithm [9] takes $O(k^2n)$ to obtain an optimal BSS in k swap passes. Clearly, a significant improvement in time complexity over nonoptimal approaches is achieved as ensuring optimality. Referring to the channel in Fig. 1, Chen's algorithm [9] only needs five swap passes to complete the connection of all the routing nets in two routing layers. However, the time complexity of Chaudhary's algorithm and Chen's algorithm is in $O(n^3)$ time in the worst case. In this paper, based on optimality-oriented swap-direction selection in an optimal BSS, an improved optimal algorithm for a BSNMCR problem is proposed, and the time complexity of the proposed algorithm is proven to be in $O(kn)$ time and in $O(n^2)$ time in the worst case.

The rest of this paper is organized as follows: Section II contains the formulation and necessary definitions for a BSNMCR problem. In Section III, the optimality-oriented swap-direction

Manuscript received December 3, 1997; revised July 8, 1998. This paper was recommended by Associate Editor T. Yoshi.

The author is with the Computer Systems Research Center, National Chiao Tung University, Hsinchu, 30050 Taiwan, R.O.C.

Publisher Item Identifier S 0278-0070(99)01009-X.

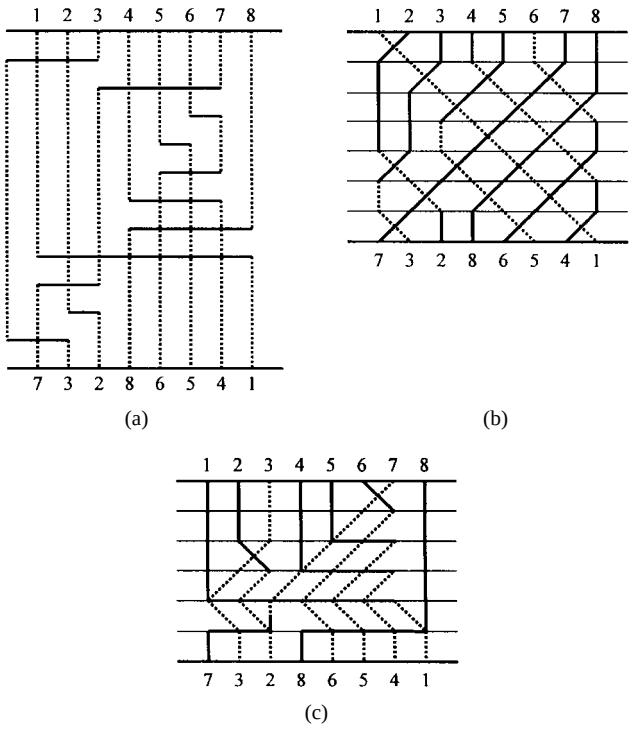

Fig. 1. Manhattan and non-Manhattan channel routing. (a) Manhattan routing result for a channel. (b) Non-Manhattan routing result by Wang's algorithm [7]. (c) Non-Manhattan routing result by Chaudhary's algorithm [8].

selection in an optimal BSS is proposed. In Section IV, an improved optimal algorithm for the two-layer BSNMCR problem is proposed and the time complexity of the proposed algorithm is further analyzed. Finally, the conclusions and further works are summarized in Section V.

II. Problem Formulation and Definitions

A channel is a rectangular routing region with two fixed terminal lists located at top and bottom boundaries in this region, respectively. A CR problem is to minimize the number of tracks for the connection of all the routing nets in a channel. According to the terminal location of the routing nets, the routing nets in a channel can be divided into *one-sided nets* and *two-sided nets*. For a one-sided net, all the terminals within this net are located on the same boundary in a channel. In contrast to the definition of a one-sided net, all the terminals within one two-sided net are located on top and bottom boundaries in a channel. On the other hand, the routing nets in a channel can be divided into *two-terminal nets* and *multiple-terminal nets* according to the number of terminals in one net.

For a BSNMCR problem, the terminals of all the routing nets are located on top and bottom boundaries in a channel. By introducing dummy and duplicated terminals into a channel [8] and renaming the terminal numbers of all the two-sided two-terminal nets in a channel, all the one-sided nets and multiple-terminal nets will be transformed into two-terminal two-sided nets. Hence, it is assumed that all the routing nets in the BSNMCR problem are two-sided two-terminal nets. For simplicity, according to the terminal order from left to right on top boundary, all the routing nets in a channel are renumbered from 1 to n, where n is the number of two-sided two-terminal nets in a channel.

In Fig. 2, one-sided nets and multiple-terminal nets in a channel are transformed into two-sided two-terminal nets by introducing dummy and duplicated terminals and renaming the terminal numbers of all the two-sided two-terminal nets. In this channel, nets 1 and 4 are one-sided two-terminal nets and introduce two dummy terminals on top and bottom boundaries, respectively. Nets 2 and 3 are two-sided three-terminal nets and introduce one duplicated terminal on top and bottom boundaries, respectively. After that, all the two-sided two-terminal nets in this channel are renamed from 1 to 11, and the resultant channel is used as the specification of the BSNMCR problem.

In general, two terminal lists on top and bottom boundaries in a channel are represented as two vectors TV and BV, respectively. Hence, TV is $(1, 2, \cdots, n)$ and BV is obtained according to the connection of all the nets. Refer to the channel in Fig. 1, $TV = (1,2,3,4,5,6,7,8)$ and $BV = (7,3,2,8,6,5,4,1)$. Therefore, the BSNMCR problem will correspond to the problem of transforming the vector BV into the vector TV pass by pass in a bubble sorting solution.

Consider a vector V, $(a_1, a_2, \cdots, a_n)$ for a BSS, two neighboring elements a_i and a_{i+1} are not in a proper order if $a_i > a_{i+1}$, for $1 \leq i < n$, and V is completely sorted if all the elements in V are in a proper order. Thus, if two neighboring elements are not in a proper order, the two elements will be swapped in a BSS. In general, if a sequence of swapping operations is considered in V from right to left, the smallest number in V will be moved toward the left. Such a sequence of right-to-left swapping operations will be defined as *a left-swap pass* for V. If V is completely sorted by a sequence of left-swap passes, the sorting solution will be defined as a *left bubble-sorting solution (LBSS)*. Similarly, a sequence of left-to-right swapping operations will be defined as a *right-swap pass* for V. If V is completely sorted by a sequence of right-swap passes, the sorting solution will be defined as a *right bubble-sorting solution (RBSS)*. For two vectors V_i and V_j, V_i is defined as a *left (right) adjacent vector* of V_j if V_i is obtained by applying a left-swap (right-swap) pass to V_j. Hence, an LBSS (RBSS) can be represented by a sequence of left (right) adjacent vectors, $V_0, V_1, \cdots, V_k$, such that $V_0 = BV$ and $V_k = TV$.

Refer to the channel in Fig. 1, an LBSS is represented by a sequence of left adjacent vectors, $V_0 = (7,3,2,8,6,5,4,1)$, $V_1 = (1,7,3,2,8,6,5,4)$, $V_2 = (1,2,7,3,4,8,6,5)$, $V_3 = (1,2,3,7,4,5,8,6)$, $V_4 = (1,2,3,4,7,5,6,8)$, $V_5 = (1,2,3,4,5,7,6,8)$, and $V_6 = (1,2,3,4,5,6,7,8)$. An RBSS is represented by a sequence of right adjacent vectors, $V_0 = (7,3,2,8,6,5,4,1)$, $V_1 = (3,2,7,6,5,4,1,8)$, $V_2 = (2,3,6,5,4,1,7,8)$, $V_3 = (2,3,5,4,1,6,7,8)$, $V_4 = (2,3,4,1,5,6,7,8)$, $V_5 = (2,3,1,4,5,6,7,8)$, $V_6 = (2,1,3,4,5,6,7,8)$, and $V_7 = (1,2,3,4,5,6,7,8)$. For the vector (7, 3, 2, 8, 6, 5, 4, 1), an LBSS needs six left-swap passes and an RBSS needs seven right-swap passes.

As mentioned in Chen's algorithm [9], if a left-swap pass and a right-swap pass are allowed to be applied to a BSS at the same time, a BSS with minimum swap passes is defined as *an optimal BSS*. For a BSNMCR problem in two-layer routing model, each swap pass in an optimal BSS is implemented by one routing track in a bubble-sorting-based non-Manhattan channel. All the routing nets in the channel are connected

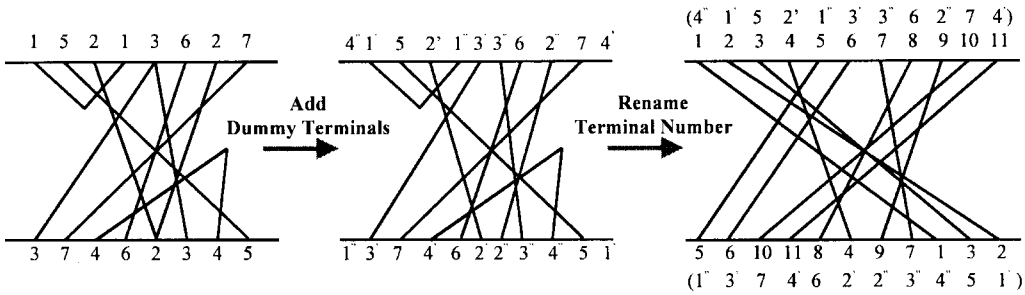

Fig. 2. Channel transformation for one-sided nets and multiple-terminal nets.

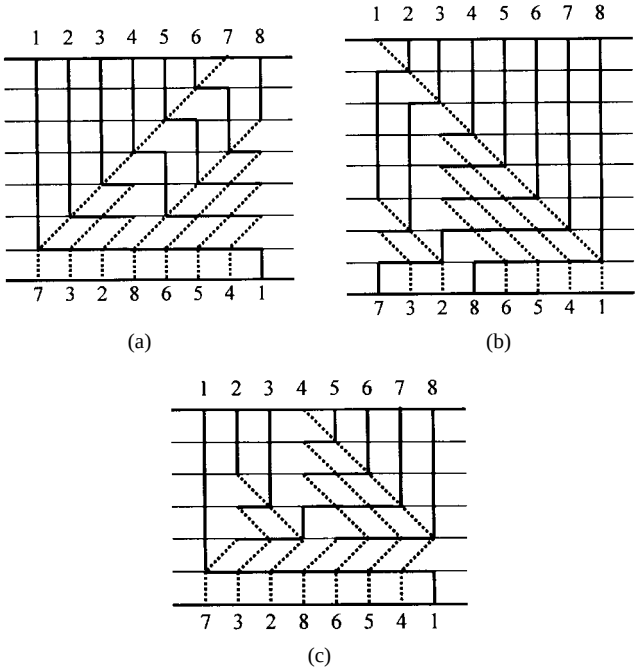

Fig. 3. Routing results for (7, 3, 2, 8, 6, 5, 4, 1). (a) Result by an LBSS. (b) Result by an RBSS. (c) Result by an optimal BSS.

by using vertical, horizontal, $+45°$ or $-45°$ wires. Hence, the number of routing tracks in a bubble-sorting-based non-Manhattan channel is equal to the number of swap passes in an optimal BSS. It is clear that the BSNMCR problem in two-layer routing model corresponds to the problem of finding an optimal BSS, that is, a minimal sequence of vectors $V_0, V_1, \cdots, V_k$ such that $V_0 = BV$, $V_k = TV$, and V_{i+1} is a left or right adjacent vector of V_i, for $0 \leq i < k$.

Refer to the channel in Fig. 1, an optimal BSS is represented by a minimal sequence of left or right adjacent vectors, $V_0 = (7,3,2,8,6,5,4,1)$, $V_1 = (3,2,7,6,5,4,1,8)$, $V_2 = (1,3,2,7,6,5,4,8)$, $V_3 = (1,2,3,4,7,6,5,8)$, $V_4 = (1,2,3,4,5,7,6,8)$, and $V_5 = (1,2,3,4,5,6,7,8)$. By using both left-swap passes and right-swap passes on (7, 3, 2, 8, 6, 5, 4, 1), an optimal BSS only needs five swap passes, that is, only five routing tracks are used to route the channel. Fig. 3 illustrates the routing results in an LBSS, an RBSS, and an optimal BSS for the channel in Fig. 1, respectively.

According to the formulation of the BSNMCR problem, it is clear that minimizing the number of routing tracks in a bubble-sorting-based non-Manhattan channel will correspond to finding an optimal BSS for a corresponding vector. Since an optimal BSS is to minimize the number of swap passes, an optimal BSS will depend on an optimality-oriented swap-direction selection. In the following, some necessary definitions in an optimality-oriented swap-direction selection are described for finding an optimal BSS.

Definition 1: For a given vector $(a_1, a_2, \cdots, a_n)$, if $i < j$ and $a_i > a_j$, then one pair (a_i, a_j) is called as *an inversion* in $(a_1, a_2, \cdots, a_n)$. Furthermore, one table $(l_1, l_2, \cdots, l_n)$ is defined as *a left inversion table* in $(a_1, a_2, \cdots, a_n)$ if the value l_i in $(l_1, l_2, \cdots, l_n)$ is the number of elements that are greater than i to the left of i, for $1 \leq i \leq n$. Similarly, one table $(r_1, r_2, \cdots, r_n)$ is defined as *a right inversion table* in $(a_1, a_2, \cdots, a_n)$ if the value r_i in $(r_1, r_2, \cdots, r_n)$ is the number of elements that are smaller than i to the right of i, for $1 \leq i \leq n$.

For a given vector $(a_1, a_2, \cdots, a_n)$, the left and right inversion tables, $(l_1, l_2, \cdots, l_n)$ and $(r_1, r_2, \cdots, r_n)$, are unique, and the values $l_1, l_2, \cdots, l_n$ in $(l_1, l_2, \cdots, l_n)$ and the values $r_1, r_2, \cdots, r_n$ in $(r_1, r_2, \cdots, r_n)$ roughly reflect the number of swap passes in an LBSS or RBSS. As a left-swap pass is applied to $(a_1, a_2, \cdots, a_n)$, the vector $(a_1, a_2, \cdots, a_n)$ will be modified by the left-swapping operations, and a new right inversion table will be obtained by decreasing all the nonzero values in $(r_1, r_2, \cdots, r_n)$ by one. Similarly, as a right-swap pass is applied to $(a_1, a_2, \cdots, a_n)$, the vector $(a_1, a_2, \cdots, a_n)$ will be modified by the right-swapping operations, and a new left inversion table will be obtained by decreasing all the nonzero values in $(l_1, l_2, \cdots, l_n)$ by one.

Definition 2: For a given vector $(a_1, a_2, \cdots, a_n)$, let $(l_1, l_2, \cdots, l_n)$ and $(r_1, r_2, \cdots, r_n)$ be the left and right inversion table of $(a_1, a_2, \cdots, a_n)$, respectively. Furthermore, the maximal left inversion value $l_{\max}$ in $(l_1, l_2, \cdots, l_n)$ will be defined as $l_{\max} = \text{Max}\{l_1, l_2, \cdots, l_n\}$, and the maximal left element $a_{\text{max-left}}$ in $(a_1, a_2, \cdots, a_n)$ will be defined as one element in $(a_1, a_2, \cdots, a_n)$ with the left inversion value $l_{\max}$. Similarly, the maximal right inversion value $r_{\max}$ in $(r_1, r_2, \cdots, r_n)$ will be defined as $r_{\max} = \text{Max}\{r_1, r_2, \cdots, r_n\}$, and the maximal right element $a_{\text{max-right}}$ in $(a_1, a_2, \cdots, a_n)$ will be defined as one element in $(a_1, a_2, \cdots, a_n)$ with the right inversion value $r_{\max}$.

Referring to the channel in Fig. 1, considering the vector (7, 3, 2, 8, 6, 5, 4, 1), the left and right inversion tables are (7, 2, 1, 4, 3, 2, 0, 0) and (0, 1, 2, 1, 2, 3, 6, 4), respectively. Clearly, the maximal left inversion value $l_{\max}$ is 7 and the maximal right inversion value $r_{\max}$ is 6.

For a complete sorted vector $(1, 2, \cdots, n)$, both the left and right inversion tables are $(0, 0, \cdots, 0)$. Considering a

BSS for $(a_1, a_2, \cdots, a_n)$, the number of swap passes depends on the transformation of the left and right inversion tables from $(l_1, l_2, \cdots, l_n)$ and $(r_1, r_2, \cdots, r_n)$ to $(0, 0, \cdots, 0)$ and $(0, 0, \cdots, 0)$, respectively. According to the modification of the right (left) inversion table, $(r_1, r_2, \cdots, r_n)$ $\{(l_1, l_2, \cdots, l_n)\}$, in a left-swap (right-swap) pass, the number of swap passes is $l_{\max}$ if a right-swap pass is only allowed to be applied to an RBSS, and the number of swap passes is $r_{\max}$ if a left-swap pass is only allowed to be applied to an LBSS. Hence, an upper bound of swap passes in an optimal BSS can be obtained as $\text{Min}\{l_{\max}, r_{\max}\}$. Since a left-swap pass and a right-swap pass are allowed to be applied to an optimal BSS at the same time, the swapping phenomenon of a left-swap pass and a right-swap pass will be considered to select an optimality-oriented swap direction in an optimal BSS.

Definition 3: For a given vector $(a_1, a_2, \cdots, a_n)$, if $p_1 < p_2 < \cdots < p_s\ (=n)$, $a_{p_1} < a_{p_2} < \cdots < a_{p_s}$, and there exists no i such that $p_j < i < p_{j+1}$ and $a_{p_j} < a_i < a_{p_{j+1}}$, for $1 \le j \le s-1$, then one list $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ is defined as *a left-swap list* in $(a_1, a_2, \cdots, a_n)$. Similarly, if $q_1(=1) < q_2 < \cdots < q_t, a_{q_1} < a_{q_2} < \cdots < a_{q_t}$, and there exists no i such that $q_j < i < q_{j+1}$ and $a_{q_j} < a_i < a_{q_{j+1}}$, for $1 \le j \le t-1$, then one list $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ is defined as *a right-swap list* in $(a_1, a_2, \cdots, a_n)$.

For the vector (7, 3, 2, 8, 6, 5, 4, 1), a vector (1, 3, 2, 7, 6, 5, 4, 8) is obtained by using a left-swap pass and a right-swap pass. Consider the vector (1, 3, 2, 7, 6, 5, 4, 8), a left-swap list (a_1, a_3, a_7, a_8) is (1, 2, 4, 8), and a right-swap list (a_1, a_2, a_4, a_8) is (1, 3, 7, 8).

According to the definitions of a left-swap (right-swap) list in $(a_1, a_2, \cdots, a_n)$, it is clear that the elements in a left-swap (right-swap) list are swapped from right to left (from left to right) in a left-swap (right-swap) pass. In a left-swap (right-swap) list $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ $((a_{q_1}, a_{q_2}, \cdots, a_{q_t}))$, $a_{p_i}(a_{q_i})$ is swapped from right to left (from left to right) until $a_{p_{i-1}}(a_{q_{i+1}})$ occurs. For a left-swap (right-swap) pass, the number of the swapping operations in $a_{p_i}(a_{q_i})$ will be defined as a *left-swap (right-swap) distance* $D_{\text{left}}(a_{p_i})$ $(D_{\text{right}}(a_{q_i}))$.

Definition 4: For a given vector $(a_1, a_2, \cdots, a_n)$, let $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ be a left-swap list and a right-swap list of $(a_1, a_2, \cdots, a_n)$, respectively. The *left-swap distance* $D_{\text{left}}(a_{p_i})$ of a_{p_i} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ is defined as

$$D_{\text{left}}(a_{p_i}) = \begin{cases} p_i - p_{i-1} - 1, & \text{for } 1 < i \le s. \\ p_i - 1, & \text{for } i = 1. \end{cases}$$

Similarly, the *right-swap distance* $D_{\text{right}}(a_{q_i})$ of a_{q_i} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ is defined as

$$D_{\text{right}}(a_{q_i}) = \begin{cases} q_{i+1} - q_i - 1, & \text{for } 1 \le i < t. \\ n - q_i, & \text{for } i = t. \end{cases}$$

Consider the vector (1, 3, 2, 7, 6, 5, 4, 8), the left-swap distances $D_{\text{left}}(1)$, $D_{\text{left}}(2)$, $D_{\text{left}}(4)$, and $D_{\text{left}}(8)$ of the left-swap list (1, 2, 4, 8) are 0, 1, 3, and 0, respectively, and the right-swap distances $D_{\text{right}}(1)$, $D_{\text{right}}(3)$, $D_{\text{right}}(7)$, and $D_{\text{right}}(8)$ of the right-swap list (1, 3, 7, 8) are 0, 1, 3, and 0, respectively.

Furthermore, for the element $a_{p_i}(a_{q_i})$ in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ $((a_{q_1}, a_{q_2}, \cdots, a_{q_t}))$, the number of such elements a_j, which are for $j < p_{i-1}$ and $a_{p_{i-1}} < a_j < a_{p_i}$ $(j > q_{i+1}$ and $a_{q_i} < a_j < a_{p_{i+1}})$ will be defined as a *left-swap (right-swap) feedback value* $F_{\text{left}}(a_{p_i})$ $(F_{\text{right}}(a_{q_i}))$ of $a_{p_i}(a_{q_i})$.

Definition 5: For a given vector $(a_1, a_2, \cdots, a_n)$, let $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ be a left-swap list and a right-swap list of $(a_1, a_2, \cdots, a_n)$, respectively. The *left-swap feedback value* $F_{\text{left}}(a_{p_i})$ of a_{p_i} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ is defined as

$$F_{\text{left}}(a_{p_i}) = \begin{cases} a_{p_i} - a_{p_{i-1}} - 1, & \text{for } 1 < i \le s. \\ 0, & \text{for } i = 1. \end{cases}$$

Similarly, the *right-swap feedback value* $F_{\text{right}}(a_{q_i})$ of a_{q_i} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ is defined as

$$F_{\text{right}}(a_{q_i}) = \begin{cases} a_{q_{i+1}} - a_{q_i} - 1, & \text{for } 1 \le i < t. \\ 0, & \text{for } i = t. \end{cases}$$

Consider the vector (1, 3, 2, 7, 6, 5, 4, 8), the left-swap feedback values $F_{\text{left}}(1)$, $F_{\text{left}}(2)$, $F_{\text{left}}(4)$, and $F_{\text{left}}(8)$ of the left-swap list (1, 2, 4, 8) are 0, 0, 1, and 3, respectively, and the right-swap distances $F_{\text{right}}(1)$, $F_{\text{right}}(3)$, $F_{\text{right}}(7)$, and $F_{\text{right}}(8)$ of the right-swap list (1, 3, 7, 8) are 1, 3, 0, and 0, respectively.

III. Swap-Direction Selection in an Optimal Bubble-Sorting Solution

For a given vector V, one vector generated by applying one left-swap pass in a BSS to V is represented as LV. Similarly, one vector generated by applying one right-swap pass in a BSS to V is represented as RV. Furthermore, one vector generated by applying a sequence of k left-swap passes in a bubble-sort solution to V is represented as L^kV, i.e., $\underbrace{LLL\cdots LLV}_{k} = L^kV$. Similarly, one vector generated by applying a sequence of k right-swap passes in a BSS to V is represented as R^kV, i.e., $\underbrace{RRR\cdots RRV}_{k} = R^kV$.

Basically, an m-pass BSS S completes the sorting process by applying a sequence $D_mD_{m-1}\cdots D_2D_1$ of left-swap and right-swap passes to V, where $D_i = L$ or R, for $1 \le i \le m$. Hence, an m-pass BSS S is represented as $S = D_mD_{m-1}\cdots D_2D_1$. To simplify the representation of a BSS, Theorem 1 in Chen's paper [9] is applied in this paper and formulated as Lemma 1.

Lemma 1: Given a vector V, the vector generated by applying one left-swap pass followed by one right-swap pass to V and the vector generated by applying one right-swap pass followed by one left-swap pass to V are the same, i.e., $LRV = RLV$.

Proof: The proof is done in [9]. □

By using the result $LRV = RLV$, a BSS S is represented as a sequence $LL\cdots LRR\cdots R$ of left-swap and right-swap passes to V. Hence, any m-pass BSS S can be further represented as L^iR^{m-i}, for $0 \le i \le m$. For an m-pass BSS S, the number of left-swap (right-swap) passes is uniquely determined if the number of right-swap (left-swap) passes in S is known. Two m-pass BSS's, $S1$ and $S2$, are equivalent if they have the same number of left-swap (right-swap) passes in $S1$ and $S2$. Clearly, the sorting result only depends on the number of left-swap passes or right-swap passes rather than

the permutation of the left-swap passes and right-swap passes in a BSS.

As mentioned above, as a left-swap pass is applied to $(a_1, a_2, \cdots, a_n)$, the vector $(a_1, a_2, \cdots, a_n)$ will be modified by the left-swapping operations, and a new right inversion table will be obtained by decreasing all the nonzero values in $(r_1, r_2, \cdots, r_n)$ by one. Similarly, as a right-swap pass is applied to $(a_1, a_2, \cdots, a_n)$, the vector $(a_1, a_2, \cdots, a_n)$ will be modified by the right-swapping operations, and a new left inversion table will be obtained by decreasing all the nonzero values in $(l_1, l_2, \cdots, l_n)$ by one. Hence, the inversion values $l_{\max}$ and $r_{\max}$ of $a_{\text{max-left}}$ and $a_{\text{max-right}}$ can be applied to decide an optimality-oriented swap direction in an optimal BSS.

Lemma 2: For a given vector $(a_1, a_2, \cdots, a_n)$, let $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ be a left-swap list and a right-swap list of $(a_1, a_2, \cdots, a_n)$, respectively. The element $a_{\text{max-left}}$ in $(a_1, a_2, \cdots, a_n)$ will be included in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and the element $a_{\text{max-right}}$ in $(a_1, a_2, \cdots, a_n)$ will be included in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$.

Proof: For a vector $(a_1, a_2, \cdots, a_n)$, assume that $a_{\text{max-left}}$ in $(a_1, a_2, \cdots, a_n)$ is not included in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$. Let $a_i = a_{\text{max-left}}$, i is less than p_1, or i is less than p_{j+1} and i is greater than p_j, i.e., $i < p_1$ or $p_j < i < p_{j+1}$ for $0 < j < s$. By the definition of a left-swap list, it is clear that $a_i > a_{p_{j+1}}$. By the definition of a left inversion table, since $i < p_{j+1}$ and $a_i > a_{p_{j+1}}$, lp_{j+1} is greater than l_i. Clearly, $a_i \neq a_{\text{max-left}}$. This is a contradiction. Hence, the element $a_{\text{max-left}}$ in $(a_1, a_2, \cdots, a_n)$ will be included in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$. Similarly, the element $a_{\text{max-right}}$ in $(a_1, a_2, \cdots, a_n)$ will be included in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$. □

Lemma 3: For a given vector $(a_1, a_2, \cdots, a_n)$, let $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ be a left-swap list and a right-swap list of $(a_1, a_2, \cdots, a_n)$, respectively. The *left inversion value* $\text{Val}_{\text{left}}(a_{p_i})$ of a_{p_i} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ is obtained as

$$\text{Val}_{\text{left}}(a_{p_i}) = \sum_{j=1}^{i} \left[D_{\text{left}}(a_{p_j}) - F_{\text{left}}(a_{p_j}) \right]$$

where $D_{\text{left}}(a_{p_j})$ and $F_{\text{left}}(a_{p_j})$ are the left-swap distance and the left-swap feedback of a_{p_j} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$, respectively. On the other hand, the *right inversion value* $\text{Val}_{\text{right}}(a_{q_i})$ of a_{q_i} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ is obtained as

$$\text{Val}_{\text{right}}(a_{q_i}) = \sum_{j=i}^{t} \left[D_{\text{right}}(a_{q_j}) - F_{\text{right}}(a_{q_j}) \right]$$

where $D_{\text{right}}(a_{q_j})$ and $F_{\text{right}}(a_{q_j})$ are the right-swap distance and the right-swap feedback of a_{q_j} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$, respectively.

Proof: For a vector $(a_1, a_2, \cdots, a_n)$, let $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ be a left-swap list and a right-swap list of $(a_1, a_2, \cdots, a_n)$, respectively. For the inversion values in a left-swap list, by the definitions of the swap distance and the swap feedback of any left-swap element, it is clear that

$$\begin{aligned}
&i = 1 \quad \text{Val}_{\text{left}}(a_{p_1}) = D_{\text{left}}(a_{p_1}) - F_{\text{left}}(a_{p_1}),\\
&i = 2 \quad \text{Val}_{\text{left}}(a_{p_2}) = \text{Val}_{\text{left}}(a_{p_1}) + D_{\text{left}}(a_{p_2}) - F_{\text{left}}(a_{p_2}),\\
&\vdots \qquad \vdots\\
\text{and} \quad &i = s \quad \text{Val}_{\text{left}}(a_{p_s}) = \text{Val}_{\text{left}}(a_{p_{s-1}}) + D_{\text{left}}(a_{p_s}) - F_{\text{left}}(a_{p_s}).
\end{aligned}$$

Therefore, the left inversion value $\text{Val}_{\text{left}}(a_{p_i})$ of a_{p_i} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ is obtained as

$$\text{Val}_{\text{left}}(a_{p_i}) = \sum_{j=1}^{i} \left[D_{\text{left}}(a_{p_j}) - F_{\text{left}}(a_{p_j}) \right]$$

where $D_{\text{left}}(a_{p_j})$ and $F_{\text{left}}(a_{p_j})$ are the left-swap distance and the left-swap feedback of a_{p_j} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$, respectively.

For the inversion values in a right-swap list, by the definitions of the swap distance and the swap feedback of any right-swap element, it is clear that

$$\begin{aligned}
&i = t \qquad \text{Val}_{\text{right}}(a_{q_t}) = D_{\text{right}}(a_{q_t}) - F_{\text{right}}(a_{q_t}),\\
&i = t-1 \quad \text{Val}_{\text{right}}(a_{q_{t-1}}) = \text{Val}_{\text{right}}(a_{q_t}) + D_{\text{right}}(a_{q_{t-1}}) - F_{\text{right}}(a_{q_{t-1}}),\\
&\vdots \qquad \vdots\\
\text{and} \quad &i = 1 \qquad \text{Val}_{\text{right}}(a_{q_1}) = \text{Val}_{\text{right}}(a_{q_1}) + D_{\text{right}}(a_{q_2}) - F_{\text{right}}(a_{q_2}).
\end{aligned}$$

Therefore, the right inversion value $\text{Val}_{\text{right}}(a_{q_i})$ of a_{q_i} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ is obtained as

$$\text{Val}_{\text{right}}(a_{q_i}) = \sum_{j=i}^{t} \left[D_{\text{right}}(a_{q_j}) - F_{\text{right}}(a_{q_j}) \right]$$

where $D_{\text{right}}(a_{q_j})$ and $F_{\text{right}}(a_{q_j})$ are the right-swap distance and the right-swap feedback of a_{q_j} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$, respectively. □

By Lemma 3, the inversion values in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ are obtained by computing the swap distances and the swap feedback of all the elements in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$. By Lemma 2, $l_{\max}$ and $r_{\max}$ are obtained from the inversion values in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$. Hence, the algorithm *MAX_Value* is designed to obtain a maximal left inversion value $l_{\max}$ and a maximal right inversion value $r_{\max}$ in $(a_1, a_2, \cdots, a_n)$ as shown at the bottom of the next page.

Theorem 1: For a vector $(a_1, a_2, \cdots, a_n)$, the algorithm *Max_Value* obtains the maximal left inversion value $l_{\max}$ and the maximal right inversion value $r_{\max}$, and the time complexity of the algorithm is in $O(n)$ time, where n is the number of elements in $(a_1, a_2, \cdots, a_n)$.

Proof: For a vector $(a_1, a_2, \cdots, a_n)$, by Lemma 2, the element $a_{\text{max-left}}$ in $(a_1, a_2, \cdots, a_n)$ is included in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and the element $a_{\text{max-right}}$ in $(a_1, a_2, \cdots, a_n)$ is included in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$. By Lemma 3, the left inversion value $\text{Val}_{\text{left}}(a_{p_i})$ of the element

a_{p_i} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ is obtained as $\mathrm{Val}_{\mathrm{left}}(a_{p_i}) = \sum_{j=i}^{t}[D_{\mathrm{left}}(a_{p_j}) - F_{\mathrm{left}}(a_{p_j})]$, where $D_{\mathrm{left}}(a_{p_j})$ and $F_{\mathrm{left}}(a_{p_j})$ are the left-swap distance and the left-swap feedback of a_{p_j} in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$, and the right inversion value $\mathrm{Val}_{\mathrm{right}}(a_{q_i})$ of a_{q_i} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ is obtained as $\mathrm{Val}_{\mathrm{right}}(a_{q_i}) = \sum_{j=1}^{i}[D_{\mathrm{right}}(a_{q_j}) - F_{\mathrm{right}}(a_{q_j})]$, where $D_{\mathrm{right}}(a_{q_j})$ and $F_{\mathrm{right}}(a_{q_j})$ are the right-swap distance and the right-swap feedback of a_{q_j} in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$. Hence, the algorithm *Max_Value* obtains the maximal right inversion value $l_{\max}$ and the maximal right inversion value $r_{\max}$ in $(a_1, a_2, \cdots, a_n)$.

For the analysis of time complexity in *Max_Value*, the step of finding a left-swap list $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and a right-swap list $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ in $(a_1, a_2, \cdots, a_n)$ takes $O(n)$ time, the step of finding all the left-swap distances $D_{\mathrm{left}}(a_{p_i})$ and all the left-swap feedback $F_{\mathrm{left}}(a_{p_i})$ in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ takes $O(n)$ time, and the step of finding all the right-swap distances $D_{\mathrm{right}}(a_{q_i})$ and all the right-swap feedback $F_{\mathrm{right}}(a_{q_i})$ in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ takes $O(n)$ time, where n is the number of elements in $(a_1, a_2, \cdots, a_n)$. Furthermore, the step to compute all the inversion values $\mathrm{Val}_{\mathrm{left}}(a_{p_i}) = \sum_{j=1}^{i}[D_{\mathrm{left}}(a_{p_j}) - F_{\mathrm{left}}(a_{p_j})]$ in $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ and all the inversion values $\mathrm{Val}_{\mathrm{right}}(a_{q_i}) = \sum_{j=i}^{t}[D_{\mathrm{right}}(a_{q_j}) - F_{\mathrm{right}}(a_{q_j})]$ in $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ takes $O(n)$ time. Finally, the step of finding the element with $l_{\max}$ and the element with $r_{\max}$ takes $O(n)$ time in the worst case. Therefore, the time complexity of the algorithm is in $O(n)$ time, where n is the number of elements in $(a_1, a_2, \cdots, a_n)$. □

Based on the computation of the inversion values $l_{\max}$ and $r_{\max}$ in $(a_1, a_2, \cdots, a_n)$, the selection of an optimality-oriented swap-direction in an optimal BSS is further formulated.

Lemma 4: For a given vector $(a_1, a_2, \cdots, a_n)$, let $(l_1, l_2, \cdots, l_n)$ and $(r_1, r_2, \cdots, r_n)$ be the left inversion table and the right inversion table of $(a_1, a_2, \cdots, a_n)$, respectively. Two conditions are used to decide an optimality-oriented swap-direction in an optimal BSS as follows.

1) If $l_{\max} > r_{\max}$, at least one left-swap pass L is included in an optimal BSS.
2) If $l_{\max} < r_{\max}$, at least one right-swap pass R is included in an optimal BSS.

Proof: For a vector $(a_1, a_2, \cdots, a_n)$, an optimal BSS is represented as $L^x R^y$, where x is the number of left-swap passes and y is the number of right-swap passes in an optimal BSS. For the proof in Condition (1), assume that if $l_{\max} > r_{\max}$, no left-swap pass L is included in an optimal BSS. Hence, an optimal BSS $S1$ is $R^{l_{\max}}$, i.e., $S1 = R^{l_{\max}}$. However, it is clear that there exists a BSS $S2 = L^{r_{\max}}$. Since $l_{\max} > r_{\max}$, the solution $S2$ will have fewer passes than the solution $S1$. Clearly, the solution $S1$ is not an optimal BSS. This is a contradiction. Therefore, if $l_{\max} > r_{\max}$, at least one left-swap pass L is included in an optimal BSS. Similarly, for the proof in Condition (2), if $l_{\max} < r_{\max}$, at least one right-swap pass R is included in an optimal BSS. □

IV. Bubble-Sorting-Based Non-Manhattan Channel Routing

For a BSNMCR problem, the routing process is based on an optimal BSS. If an optimal BSS is obtained, all the swapping operations in each pass will be implemented by routing vertical, horizontal, $+45°$ or $-45°$ wires. Hence, the BSNMCR problem is divided into *optimal bubble-sorting solution* and *bubble-sorting-based non-Manhattan channel routing*.

A. Optimal Bubble-Sorting Solution

For a vector $(a_1, a_2, \cdots, a_n)$, $l_{\max}$ and $r_{\max}$ are obtained from $(l_1, l_2, \cdots, l_n)$ and $(r_1, r_2, \cdots, r_n)$, respectively. By Lemma 4, there are two conditions to decide an optimality-oriented swap-direction for $(a_1, a_2, \cdots, a_n)$ in an optimal BSS. Therefore, the algorithm *OBSS* is designed to obtain an optimal BSS as shown at the bottom of the next page.

Theorem 2: The algorithm *OBSS* obtains an optimal BSS and the time complexity of the algorithm is $O(kn)$, where k is the number of swap passes in an optimal BSS and n is the number of elements in $(a_1, a_2, \cdots, a_n)$.

Proof: For a vector $(a_1, a_2, \cdots, a_n)$, by Theorem 1, the algorithm *Max_Value* obtains the maximal left inversion value $l_{\max}$ and the maximal right inversion value $r_{\max}$. By Lemma 4, based on the comparison of the values $l_{\max}$, and $r_{\max}$, a swap-direction in $(a_1, a_2, \cdots, a_n)$ is correctly decided for an optimal BSS. Hence, the algorithm *OBSS* can obtain an optimal BSS.

For the analysis of time complexity in *OBSS*, by Theorem 1, the time complexity of the algorithm *Max_Value* is in $O(n)$ time, where n is the number of elements in $(a_1, a_2, \cdots, a_n)$. In the decision of a swap-direction in $(a_1, a_2, \cdots, a_n)$, the time complexity of comparing the values $l_{\max}$ and $r_{\max}$ and assigning the values $l_{\max}$ and $r_{\max}$ is in $O(1)$ time. Since the number of swap passes in an optimal BSS is k, the number of

---

Algorithm Max_Value;
Input: *a vector* $(a_1, a_2, \cdots, a_n)$;
begin
Find a left-swap list $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$ *and a right-swap list* $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$ *in* $(a_1, a_2, \cdots, a_n)$;
Find all the left-swap distances $D_{\mathrm{left}}(a_{p_i})$ *and all the left-swap feedback* $F_{\mathrm{left}}(a_{p_i})$ *in* $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$;
Find all the right-swap distances $D_{\mathrm{right}}(a_{q_i})$ *and all the right-swap feedback* $F_{\mathrm{right}}(a_{q_i})$ *in* $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$;
Compute all the inversion values $\mathrm{Val}_{\mathrm{left}}(a_{p_i}) = \sum_{j=1}^{i}[D_{\mathrm{left}}(a_{p_j}) - F_{\mathrm{left}}(a_{p_j})]$ *in* $(a_{p_1}, a_{p_2}, \cdots, a_{p_s})$
and all the inversion values $\mathrm{Val}_{\mathrm{right}}(a_{q_i}) = \sum_{j=i}^{t}[D_{\mathrm{right}}(a_{q_j}) - F_{\mathrm{right}}(a_{q_j})]$ *in* $(a_{q_1}, a_{q_2}, \cdots, a_{q_t})$;
Output the maximal right inversion value $l_{\max}$ *and the maximal right inversion value* $r_{\max}$;
end

iterations in this loop will be k. Therefore, the time complexity of the algorithm *OBSS* is $O(kn)$, where k is the number of swap passes in an optimal BSS and n is the number of elements in $(a_1, a_2, \cdots, a_n)$. □

Referring to the channel in Fig. 1, (7, 3, 2, 8, 6, 5, 4, 1) is transformed into (1, 2, 3, 4, 5, 6, 7, 8) pass by pass in a BSS. By running the algorithm *OBSS*, an optimal BSS is obtained by the passes shown at the bottom of the page.

For $V = (7,3,2,8,6,5,4,1)$, an optimal BSS is $LLLRLV = (1,2,3,4,5,6,7,8)$. Hence, an optimal BSS S is L^4R, i.e., $S = L^4R$. If two layers are available in a bubble-sorting-based non-Manhattan routing model, by mapping each left-swap or right-swap pass in L^4R on one routing track from bottom to top track by track, a physical routing result shown will be obtained and shown in Fig. 4.

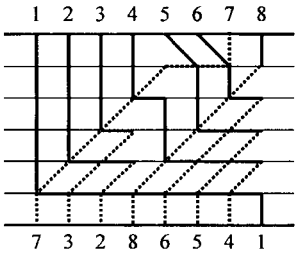

Fig. 4. Physical routing result for an optimal BSS of (7, 3, 2, 8, 6, 5, 4, 1).

B. Bubble-Sorting-Based Non-Manhattan Channel Routing

Consider bubble-sorting-based non-Manhattan channel routing, first, all the one-sided nets and multiple-terminal nets in a channel are transformed into two-terminal two-sided

---

Algorithm OBSS
Input: a vector $(a_1, a_2, \cdots, a_n)$;
begin
 while (*the vector* $(a_1, a_2, \cdots, a_n)$ *is unsorted*)
  begin
   Run the algorithm Max_Value to obtain the maximal left inversion value $l_{\max}$ *and the maximal right inversion value* $r_{\max}$;
   If $(l_{\max} > r_{\max})$, *a left-swap pass* L *is applied to* $(a_1, a_2, \cdots, a_n)$;
   If $(l_{\max} < r_{\max})$, *a right-swap pass* R *is applied to* $(a_1, a_2, \cdots, a_n)$;
   If $(l_{\max} = r_{\max})$, *a left-swap pass* L *or right-swap pass* R *is applied to* $(a_1, a_2, \cdots, a_n)$;
  end
 Output an optimal BSS;
end

Pass 1: $V = (7,3,2,8,6,5,4,1)$
Left inversion table *Right inversion table* *Left-swap list* *Right-swap list*
$(7,2,1,4,3,2,0,0)$ $(0,1,2,1,2,3,6,4)$ (1) $(7,8)$
→ Condition: $l_{\max}(=7) > r_{\max}(=6)$, Decision: a left-swap pass L to $(7,3,2,8,6,5,4,1)$
→ If a left-swap pass L to $(7,3,2,8,6,5,4,1)$, then $LV = (1,7,3,2,8,6,5,4)$

Pass 2: $LV = (1,7,3,2,8,6,5,4)$
Left inversion table *Right inversion table* *Left-swap list* *Right-swap list*
$(0,2,1,4,3,2,0,0)$ $(0,0,1,0,1,2,5,3)$ $(1,2,4)$ $(1,7,8)$
→ Condition: $l_{\max}(=4) < r_{\max}(=5)$, Decision: a right-swap pass R to $(1,7,3,2,8,6,5,4)$
→ If a right-swap pass R to $(1,7,3,2,8,6,5,4)$, then $RLV = (1,3,2,7,6,5,4,8)$

Pass 3: $RLV = (1,3,2,7,6,5,4,8)$
Left inversion table *Right inversion table* *Left-swap list* *Right-swap list*
$(0,1,0,3,2,1,0,0)$ $(0,0,1,0,1,2,3,0)$ $(1,2,4,8)$ $(1,3,7,8)$
→ Condition: $l_{\max}(=3) = r_{\max}(=3)$, Decision: a left-swap pass L or right-swap pass R to $(1,3,2,7,6,5,4,8)$
→ If a left-swap pass L to $(1,3,2,7,6,5,4,8)$, then $LRLV = (1,2,3,4,7,6,5,8)$

Pass 4: $LRLV = (1,2,3,4,7,6,5,8)$
Left inversion table *Right inversion table* *Left-swap list* *Right-swap list*
$(0,0,0,0,2,1,0,0)$ $(0,0,0,0,0,1,2,0)$ $(1,2,3,4,5,8)$ $(1,2,3,4,7,8)$
→ Condition: $l_{\max}(=2) = r_{\max}(=2)$, Decision: a left-swap pass L or right-swap pass R to $(1,2,3,4,7,6,5,8)$
→ If a left-swap pass L to $(1,2,3,4,7,6,5,8)$, then $LLRLV = (1,2,3,4,5,7,6,8)$

Pass 5: $LLRLV = (1,2,3,4,5,7,6,8)$
Left inversion table *Right inversion table* *Left-swap list* *Right-swap list*
$(0,0,0,0,0,1,0,0)$ $(0,0,0,0,0,0,1,0)$ $(1,2,3,4,5,6,8)$ $(1,2,3,4,5,7,8)$
→ Condition: $l_{\max}(=1) = r_{\max}(=1)$, Decision: a left-swap pass L or right-swap pass R to $(1,2,3,4,5,7,6,8)$
→ If a left-swap pass L to $(1,2,3,4,5,7,6,8)$, then $LLLRLV = (1,2,3,4,5,6,7,8)$

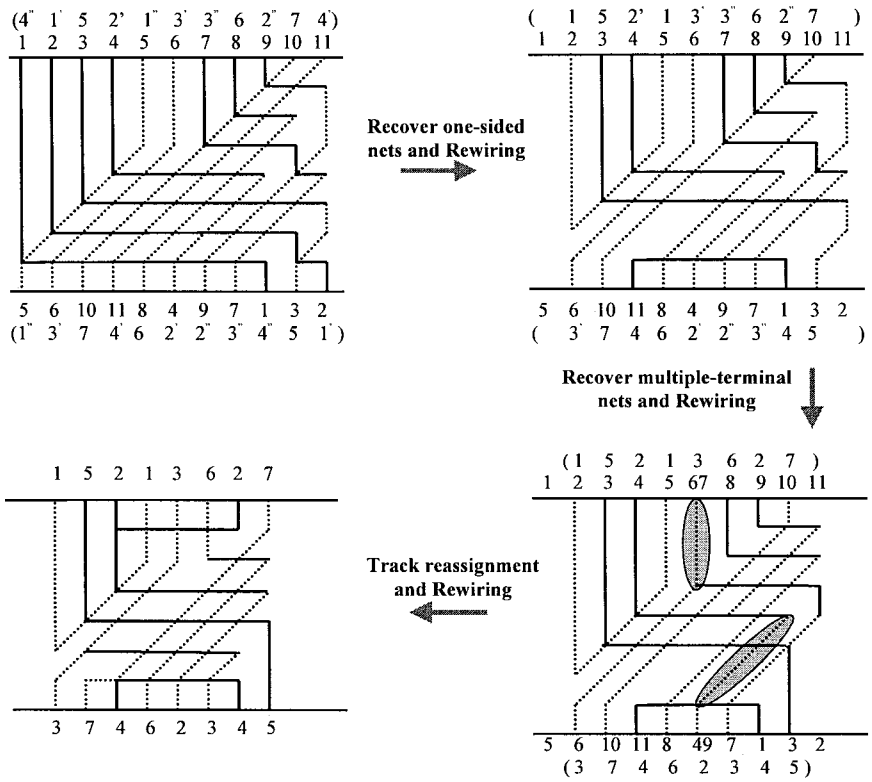

Fig. 5. Recovery of one-sided nets and multiple-terminal nets in a bubble-sorting-based non-Manhattan channel.

nets by introducing dummy and duplicated terminals into this channel and renaming all the terminal numbers of all the two-terminal two-sided nets in this channel. Refering to Fig. 2, for a channel specification, $TV = (1,5,2,1,3,6,2,7)$ and $BV = (3,7,4,6,2,3,4,5)$, nets 1 and 4 in one-sided nets introduce two dummy terminals on top and bottom boundaries, respectively. On the other hand, nets 2 and 3 in multiple-terminal nets introduce one duplicated terminal on top and bottom boundaries, respectively. Hence, a new channel specification is obtained as $TV = (4'',1',5,2',1'',3',3'',6,2'',7,4')$ and $BV = (1'',3',7,4',6,2',2'',3'',4'',5,1')$. Furthermore, by renaming all the terminal numbers of all the two-terminal two-sided nets in this channel, a final new channel specification is obtained as $TV = (1,2,3,4,5,6,7,8,9,10,11)$ and $BV = (5,6,10,11,8,4,9,7,1,3,2)$.

Basically, BSNMCR is based on an optimal BSS. If an optimal BSS is obtained by the algorithm *OBSS*, this solution will be represented as L^xR^y, where x is the number of left-swap passes and y is the number of right-swap passes in an optimal BSS, respectively. As mentioned above, BSNMCR uses the result of an optimal BSS to yield a physical routing result for a given channel. In this BSNMCR problem, all the routing nets in a channel are connected by using vertical, horizontal, $+45°$ or $-45°$ wires. If two layers are available in a bubble-sorting-based non-Manhattan routing model, a physical routing result will be yielded by mapping each left-swap or right-swap pass in L^xR^y on one routing track from bottom to top track by track. For the vector (5, 6, 10, 11, 8, 4, 9, 7, 1, 3, 2), an optimal BSS L^7 is obtained by running the algorithm *OBSS*. Furthermore, a physical routing result will be obtained by mapping seven left-swap passes in L^7 on seven tracks from bottom to top track by track.

Finally, according to a physical routing result of all the two-sided two-terminal nets in a channel, the routing of all the one-sided nets and multiple-terminal nets must be recovered from that of all the two-sided two-terminal nets in a channel. Basically, the routing of each one-sided net is obtained by eliminating dummy terminals and redundant wires and introducing vias on crossing points of two nets. After this, a rewiring process may be used to improve the routing result. On the other hand, the routing of each multiple-terminal net is obtained by merging duplicated terminals, eliminating redundant wires and introducing vias on the endpoints of merged wires. After this, a track-reassignment and rewiring process may be used to improve the routing result.

In Fig. 5, for the vector (5, 6, 10, 11, 8, 4, 9, 7, 1, 3, 2), a physical routing result has seven tracks by mapping seven left-swap passes in L^7. For the recovery of one-sided nets, nets 1 and 4 are routed by eliminating two dummy terminals and redundant wires and introducing one via on crossing points of two nets, respectively. For the recovery of multiple-terminal nets, nets 2 and 3 are routed by merging one duplicated terminal, eliminating redundant wires, and introducing one via on the endpoints of merged wires, respectively. Finally, a track-reassignment and rewiring process is used to reduce the number of routing tracks and total wire length.

V. Conclusions and Further Works

For a bubble-sorting-based non-Manhattan channel routing (BSNMCR) problem, Chaudhary's $O(kn^2)$ heuristic algorithm [8] and Chen's $O(k^2n)$ optimal algorithm [9] have been respectively proposed, where n is the number of terminals and k is the number of routing tracks in a channel. However, the

time complexity of the two algorithms is in $O(n^3)$ time in the worst case. In this paper, based on optimality-oriented swap-direction selection in an optimal BSS, an improved optimal algorithm for a BSNMCR problem is proposed, and the time complexity of the proposed algorithm is proven to be in $O(kn)$ time and in $O(n^2)$ time in the worst case.

Basically, the main goal of a BSNMCR problem is to minimize total routing area in a channel. It is clear that total routing area in a bubble-sorting-based non-Manhattan channel depends on routing modeling and the number of tracks. In further works, designing an optimal routing model for a bubble-sorting-based non-Manhattan channel and proposing an effective bubble-sorting-based approach to reduce the number of tracks will be studied for a BSNMCR problem.

REFERENCES

[1] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Design Automation Workshop*, 1971, pp. 214–224.
[2] R. L. Rivest and C. M. Fiduccia, "A greedy channel router," in *ACM/IEEE Design Automation Conf.*, 1982, pp. 418–424.
[3] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. Computer-Aided Design*, vol. 1, pp. 25–35, 1982.
[4] M. Burstein and R. Pelavin, "Hierarchical channel router," in *ACM/IEEE Design Automation Conf.*, 1983, pp. 591–597.
[5] D. N. Deutsch, "A dogleg channel router," in *ACM/IEEE Design Automation Conf.*, 1976, pp. 425–433.
[6] E. Lodi, F. Luccio, and L. Pagli, "A preliminary study of a diagonal channel-routing model," *Algorithmica*, vol. 4, pp. 585–597, 1989.
[7] D. Wang, "Novel routing schemes for IC layout—Part I: Two-layer channel routing," in *ACM/IEEE Design Automation Conf.*, 1991, pp. 49–53.
[8] K. Chaudhary and P. Robinson, "Channel routing by sorting," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 754–760, 1991.
[9] C. Y. R. Chen, C. Y. Hou, and U. Singh, "Optimal algorithms for bubble sort based non-Manhattan channel routing," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 603–609, 1994.

Jin-Tai Yan received the B.S., M.S., and Ph.D. degrees in computer and information science from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., in 1988, 1990, and 1995, respectively.

From 1995 to 1997 he served in the Chinese Navy, Kaohsiung, Taiwan, as an Information Officer working on ship information management systems. Since 1997, he has been with National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., and is currently a Post-Doctor Researcher working on CPU design and the flow control of CAD tools in the Computer Systems Research Center. His current research interests are high-level synthesis, logic synthesis, and physical design of VLSI circuits, parallel/distributed computing, interconnection networks, and network performance analysis.