ing the leaves. Hence, the time required is affected significantly by the depth of the structure. If the communication delay (t_p) is relatively large (hence a larger k), it will be more costly to propagate a message in a skinny structure than in a bushy structure. This result suggests that we may improve the performance by manipulating the structure of the knowledge-base: if we have *a priori* knowledge of the communication delay, we may restructure the knowledge-base to be bushy if k is relatively large; otherwise, we may organize it to be skinnier.

For the last observation, we realize that a skinny structure has fewer branches than a bushy structure. Since our message propagation scheme generates copies of message sequentially, the time between the generation of the first and the last copies of the outgoing messages is less than that in a bushy structure. Therefore, the curve of a skinny structure tends to have a smaller slope.

VII. Concluding Remarks

In this paper, we have presented a message-oriented knowledge-based model. Both our model and the backtracking model were analyzed and compared.

Acknowledgment

We thank J. Gilbert for his comments on the earlier draft of this paper.

References

[1] L. Bic and C. Lee, "A data-driven model for a subset of logic programming," *ACM Trans. Programming Languages (TOPLAS)*, vol. 9, no. 4, Oct. 1987.
[2] *IEEE Comput. Mag.*, Special Issue on Dataflow Systems, vol. 15, no. 2, Feb. 1982.
[3] A. Deliyanni and R. A. Kowalski, "Logic and semantic networks," *Commun. ACM*, vol. 22, no. 3, pp. 184–192, 1979.
[4] R. Kowalski, *Logic for Problem Solving*. New York: North-Holland, 1979.
[5] T. J. Lehman, "Design and performance evaluation of a main memory relational database system," Ph.D. dissertation, CS Dep., Univ. of Wisconsin-Madison, 1986.
[6] W. Wong, T. Suda, and L. Bic, "Performance analysis of a message-oriented knowledge-base," TR 87-11. ICS Dep., Univ. of California, Irvine, 1987.

An Efficient Channel Routing Algorithm to Yield an Optimal Solution

JIA-SHUNG WANG AND R. C. T. LEE

Abstract—**In this paper, we propose an algorithm named OCR (optimal channel routing) which finds an optimal solution for the channel routing problem in VLSI design. Since the channel routing problem is NP-complete, we cannot expect any algorithm to have polynomial time complexity to solve this problem for worst cases. Our algorithm is an A^* algorithm with good heuristics and dominance rules to terminate unnecessary nodes in the searching tree. Experimental results show that it behaves quite well in average cases which coincides with theoretical analysis. We obtained an optimal solution for the famous Deutsch difficult case in 5.5 min CPU time after our algorithm was implemented in Pascal and run on a VAX 11/750 computer.**

Index Terms—**A^* algorithm, channel routing, Dilworth theorem, dominance rules, heuristic rules, NP-complete.**

I. Introduction

In the design of automatic wire routing of VLSI chips, a channel router is one of the basic tools. A two-layer channel consists of two rows of terminals. Each net may contain some terminals on the top and bottom rows. Between the top and bottom rows, there are some horizontal tracks such that the horizontal segments of nets can be assigned. Nets are routed with horizontal segments on one layer and vertical segments on the other. The channel routing problem is to find a feasible solution to assign nets into tracks so that the number of tracks used is minimized.

A channel routing problem can be characterized by two constraint graphs: a vertical constraint graph [5] and a horizontal constraint graph. A vertical constraint graph can be considered as a partial ordering [15] of nets which will be assigned to horizontal tracks. A horizontal constraint graph indicates possible ways of assigning nets into one track. The channel routing problem was proved to be an NP-complete problem in [9] and [16]. Thus, it is quite unlikely that an efficient algorithm exists to solve it in the worst case. But we shall show that it can be solved quickly in some restricted cases.

For a vertical constraint graph, let w be the number of elements in the largest antichain [15] (w is called the Dilworth number [3]). We denote a restricted version of the channel routing problem, whose corresponding vertical constraint graph has a bounded Dilworth number, by the *bounded channel routing problem*. We show that there exists a polynomial time algorithm for this restricted problem.

Our algorithm is an A^* algorithm [11], [12], [14] which is a tree-searching algorithm. Using the vertical constraint graph, we may determine the set of nets which may be assigned next. Since not all of the nets can be assigned into one track, we use the horizontal constraint graph to divide the candidates into groups. Each group is a node in the searching tree and consists of a set of nets which can be assigned to the same track. With a proper cost function, the algorithm is terminated and an optimal solution is obtained as soon as a leaf node is reached. We can terminate the algorithm due to the properties of the A^* algorithm and the cost evaluation function used.

In our algorithm, some dominance rules [6], [10], [13] are used to terminate the nodes which need not be expanded. A special data structure is used so that whether a node should be terminated or not can be decided in linear time. This data structure is a tree-like [8], [17] data structure.

The worst case performance of this algorithm is measured by the maximum number of nodes generated in the searching tree. We show that the maximum number of nodes which can be generated in the searching tree is $wd(d+1)^{w-1}$, where $d = \lceil N/w \rceil$, w is the Dilworth number of the vertical constraint graph corresponding to the channel routing problem, and N is the total number of nets. This means that in the worst case, the number of nodes generated is a polynomial function with respect to w. We can prove that the lower bound of the bounded channel routing problem is $\Omega(d^w)$ [17]. This means that our algorithm is near optimal. Moreover, the average performance of this algorithm is shown to be quite good. For a vertical constraint graph consisting of linear orderings (chains) only and having a bounded Dilworth number, the average number of nodes generated is $O(N)$, where N is the total number of nets.

This algorithm has been implemented in Pascal on a VAX 11/750 computer. Many testing cases have been used to test the efficiency of our algorithm. We obtained an optimal solution for the famous Deutsch difficult case [1], [2], which consists of 72 nets, in 5.5 min CPU time. We also tested our algorithm on many real world cases

Manuscript received September 30, 1987; revised November 29, 1988. This work was supported by the National Science Council of the Republic of China under Grant NSC76-0408-E007-04.

J.-S. Wang is with the Institute of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 30043, Republic of China.

R. C. T. Lee is with the National Tsing Hua University and Academic Sinica, Taipei, Taiwan, Republic of China.

IEEE Log Number 9034546.

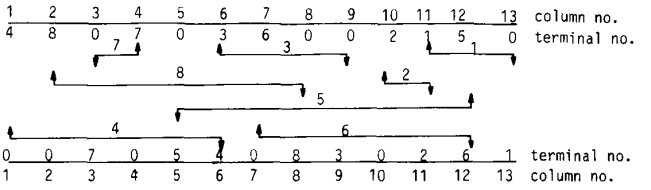

Fig. 1. A channel specification.

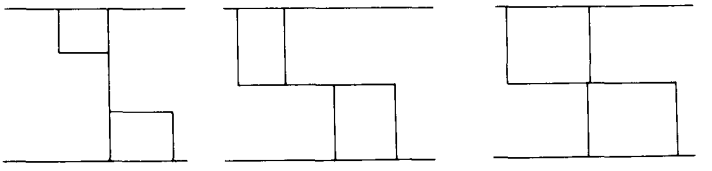

Fig. 2. Illegal wiring.

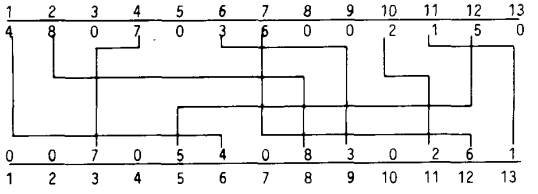

Fig. 3. An optimal layout of Fig. 1.

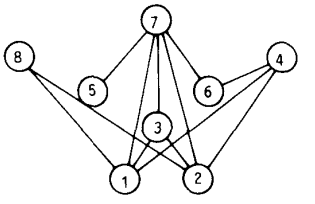

Fig. 4. The horizontal constraint graph for the channel specification of Fig. 1.

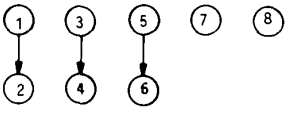

Fig. 5. The vertical constraint graph for the channel specification of Fig. 1.

from a local electronics research organization. Up to now the most difficult case, involving 60 nets, was solved in less than 2 min CPU time.

II. The Channel Routing Problem

In the channel routing problem, we are given a channel consisting of two rows (the top row and bottom row). The number of columns is the same for both rows. We are also given a set of nets. Each net connects a set of columns, and these columns are called the terminals of that net. In each channel routing problem, we are given a specification about how nets are to be connected and denote it by the *channel specification* of the problem. Consider Fig. 1. This channel specification contains eight nets of the two-terminal type. Net 1 connects column 11 of the top row with column 13 of the bottom row. Similarly, net 8 connects column 2 of the top row with column 8 of the bottom row. Between the top row and the bottom row, there exist some horizontal tracks, in which nets can be assigned under some constraints.

In the following, we use the directional layout model, where two layers are assumed. One layer is for the horizontal lines, and the other layer is for the vertical lines. Based upon this model, no two lines of different nets can overlap either horizontally or vertically as shown in Fig. 2. Furthermore, we use the wiring style which utilizes only one horizontal track per net because the number of vias needed and the length of wires are both minimum in this wiring style [2], [9].

Given a channel and a set of nets, the channel routing problem is to assign these nets into tracks in such a way that no illegal overlapping occurs and the number of tracks is minimized. It is assumed that each net occupies only one track. Fig. 3 shows a possible solution for the problem shown in Fig. 1.

III. The Horizontal and Vertical Constraints

The requirement that no illegal overlapping occurs may introduce some constraints on the channel routing problem. There are two kinds of constraints, horizontal and vertical. The horizontal constraints can be explained by considering Fig. 1 again. Net 4 cannot share the same track with net 7 because they overlap with each other horizontally. Yet, net 4 and net 6 can be put into the same track. Similarly, net 4 and net 1 can be put into the same track. We define a net x precedes a net y if the rightmost terminal of net x is located to the left of the leftmost terminal of net y. This precedence relation forms a partial ordering, and this partial ordering induces a horizontal constraint graph. Two nets can be put into the same track only if they are connected in the horizontal constraint graph. For the case in Fig. 1, the horizontal constraint graph is shown in Fig. 4. The horizontal constraint graph is a special case of the comparability graph [4].

Consider Fig. 1 again. Since net 1 is connected to column 11 of the top row and net 2 is connected to column 11 of the bottom row, there is some precedence relation between them. Note that column 11 of net 1 is on the top row. This means that net 1 must be in a track closer to the top row than the track assigned to net 2. This vertical constraint induces a vertical constraint graph [7]. A net i must be assigned before a net j if i is a predecessor of j in the vertical constraint graph. For the case in Fig. 1, its vertical constraint graph is shown in Fig. 5. This graph is composed of 5 disjoint chains since all of the nets are of the two-terminal type.

Recall that the wiring style we used requires one net uses only one track. Sometimes the vertical constraint relations among two or more nets may be cyclical which will induce a conflicting condition. However, this kind of conflicting situation can be avoided by rearranging the placement of pins or by adding new columns to the channel. We assume that the conflicting conditions are solved by a preprocessor.

A channel routing problem can then be characterized by these two constraint graphs. The vertical constraint graph informs us which nets can be considered as candidates to be put into the next track. Since they may not be put into the same track, we can consult the

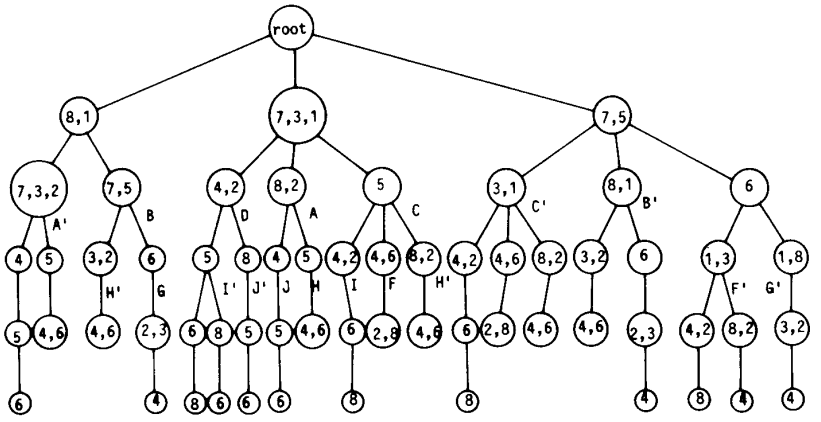

Fig. 6. A fully expanded searching tree.

horizontal constraint graph to determine how they can be broken into groups and how each group can be put into one track. Consider Fig. 4 and Fig. 5 again. In the horizontal constraint graph of $\{1,3,5,7,8\}$, there are three maximal cliques (complete subgraphs) $\{8,1\}$, $\{7,5\}$, and $\{7,3,1\}$. This indicates that there are three possible ways to put nets into the first track (nets 8 and 1), (nets 7 and 5), and (nets 7, 3, and 1).

It is critically important to note that in obtaining an optimal solution to the channel routing problem, only maximal cliques need to be considered. This will be our branching method of the proposed algorithm described in the next section. In fact, the good result of the average case performance of our proposed algorithm is mostly based upon this property.

The complexity of the channel routing problem is shown as follows. Let CS be a channel specification containing N nets, and VCG(CS) be its corresponding vertical constraint graph. Assume that VCG(CS) is composed of w disjoint chains, whose lengths are $d_1, d_2, \cdots, d_w$, and $d_1 + d_2 + \cdots + d_w = N$. For the extreme case, $w = N$, this problem can be solved by the left-edge algorithm [5] with time complexity $O(N \log N)$. For another extreme case, $w = 2$ and $d_1 = d_2$, the lower bound of this problem is $d_1 \times d_2$ [17]. We can obtain this bound by using the dynamic programming approach. In general, the lower bound of the bounded channel routing problem is $d_1 \times d_2 \times \cdots \times d_w$ [17]. The generalized dynamic programming algorithm can achieve this bound.

The straightforward dynamic programming algorithm for the bounded channel routing problem has the disadvantage of taking the same amount of time on all input cases. In practice, we prefer an algorithm which is more efficient on most of the cases but may be less efficient on some of the worst cases. This fact is the reason that we use the A^* algorithm as our search strategy.

IV. Our Proposed Algorithm

In the previous section, we showed how we can use the horizontal constraint graph and the vertical constraint graph to assign nets into tracks. In this section, we shall show how these two graphs can be used to find a solution with the smallest number of tracks.

In the channel routing problem, there is a partial ordering for the nets as described by the vertical constraint graph. Therefore, there are topological sequences of nets. Each sequence indicates the ordering of nets assigned to tracks. That is, it will correspond to a feasible solution.

A straightforward way of solving the channel routing problem is to enumerate all possible topological sequences based upon the vertical constraint graph. For each sequence, conduct a linear scan and use the horizontal constraint graph to assign nets into the same track. An optimal assignment can be found after this enumeration through a tree-searching method. The channel routing problem shown in Fig. 1 can be solved by fully expanding the tree as shown in Fig. 6. In Fig. 6, each path represents a solution and optimal solutions are solutions with the smallest number of tracks. As shown in Fig. 6, there are eight optimal solutions. The total number of tracks used in these optimal solutions is four. It is evident that a fully expanded tree represents an exhaustive searching of the entire solution space and is therefore highly inefficient. In the following paragraphs, we shall show that many branches of the tree can be pruned prematurely, and an A^* algorithm can be used to search for an optimal solution.

In the following, we shall show that the exhaustive searching can be avoided by using some dominance rules [13] to rule out those nodes which will not lead to better solutions.

Consider Fig. 6. For node A, the remaining net set $R(A)$ is $\{4,5,6\}$, and Node A', $R(A') = \{4,5,6\}$. These two nodes are on the same level of the tree and also have the same remaining net set. It is clear that the number of tracks used by an optimal solution below node A is equal to that below node A'. In such a case, either A or A' may be terminated. Therefore, we shall define three dominance rules:

Rule 1: Either node X or node Y can be terminated if X and Y are in the same level of the searching tree and they have the same remaining net set.

Rule 2: Node X should be terminated if the level of X is greater than that of node Y and they have the same remaining net set.

Rule 3: Node X should be terminated if the level of node X is greater than or equal to that of node Y, and the remaining net set $R(Y)$ is properly contained in the remaining net set $R(X)$.

In the above rules, if a node Y is terminated because of a node X, then we shall say that Y is dominated by X. Node Y is called a dominated node. If node X is not dominated by other nodes, then X is called a nondominated node.

By using the above rules, the searching tree in Fig. 6 is trimmed to that shown in Fig. 7.

Let us now introduce a term, *channel density*. Let d_i, denoted as the number of nets which pass column i, be called the local density of column i. The channel density is the maximum value of d_i, $i = 1, 2, \cdots, M$, where M is the number of columns. For the case in Fig. 1, $d_1 = 1$, $d_7 = 4$, and $d_{11} = 4$ and the channel density is 4.

There are two trivial lower bounds of the number of tracks required for wiring a channel specification (CS). The first one is the channel density cd(CS), and the other is the length of the longest path in the vertical constraint graph lp(CS). Naturally, we can combine these two lower bounds by a maximum operator into a new lower bound $\max\{\text{cd(CS)}, \text{lp(CS)}\}$. In a node v of a searching tree, the cost value $c_1(v)$ of node v is defined as the sum of two values: 1) the length of the path from the root node to node v, and 2) the estimation of the

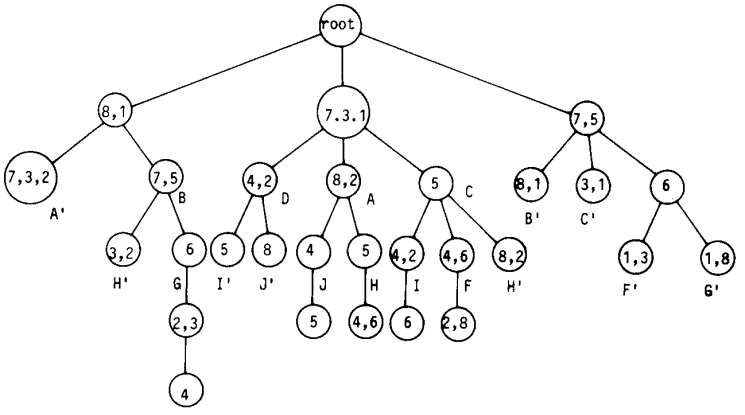

Fig. 7. The tree in Fig. 6 trimmed by using dominance rules.

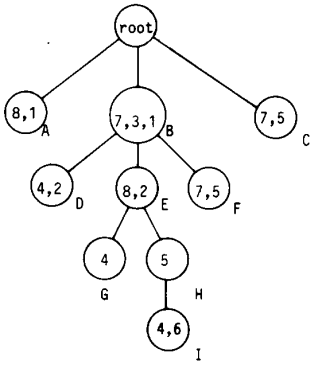

Fig. 8. A partially expanded searching tree obtained by using cost functions.

minimum tracks needed from node v to a leaf. For brevity, let $R(v)$ denote the set of nets which have not been assigned. We shall call this set the remaining net set of node v. Our cost evaluation function $c(v)$ is defined as follows:

$$c(v) = (c_1(v), c_2(v)),$$

$$c_1(v) = \text{lev}(v) + \max\{\text{rcd}(v), \text{rlp}(v)\} \text{ and } c_2(v) = \text{rn}(v)$$

where lev(v) is the level of this node on the searching tree, rcd(v) is the channel density of the set of remaining nets $R(v)$, rlp(v) is the length of the longest path in the vertical constraint graph for $R(v)$, and rn(v) is the number of nets which have not been assigned.

To compare cost values of two nodes, we use lexicographical ordering. That is, we compare the first term first. If they are equal, then compare the later. Using the cost function as described above and the termination rule, an optimal solution for the problem shown in Fig. 1 can now be found by partially expanding the searching tree as in Fig. 8.

Function $c_1(v)$ is called the major cost function and contains the actual cost value of node v. $c_2(v)$ is called the auxiliary cost function and has nothing to do with the cost value, but it gives a depth-first traversal in the partial tree with the same major cost function. That is, if node X is the currently expanded node with major cost r, and node Y is one of the sons having major cost r and the smaller auxiliary cost value, then node Y will be expanded immediately after node X. We can say that the cost function $c(v)$ prefers selecting the node in the highest level of the searching tree among all the nodes with the same least major cost value. For example, in the case of Fig. 8, nodes A, B, C, E, F, and I have the same major cost value 4. The traversal path guided by the auxiliary function is $B \rightarrow E \rightarrow H \rightarrow I$. There are some properties of the cost evaluation function which will be useful for later discussion. The following lemmas and theorems will be given without proofs. The proofs can be found in [17].

Lemma 1: If node u is a son of a node v, then $c_1(u) = c_1(v)$ or $c_1(u) = c_1(v) + 1$.

Lemma 2: If the least-cost first strategy is used and a node u is expanded immediately after node v is expanded, then $c_1(u) = c_1(v)$ or $c_1(u) = c_1(v) + 1$.

Lemma 2 shows that the major cost values of a sequence of nodes expanded by using Algorithm OCR are nondecreasing. Furthermore, for any consecutive pair of expanded nodes, the difference of their major cost values is less than two. The following theorem explains how we obtained an optimal solution upon arriving at a leaf node.

Theorem 1: Using the cost function $c(v)$ and the least-cost first strategy, the path from the top of the node to the first-found leaf node represents an optimal solution.

Algorithm OCR is listed below.

Algorithm OCR (Optimal Channel Routing)

Input: A channel specification.

Output: An optimal solution $A_1, A_2, \cdots, A_k$, where A_i is the set of nets assigned to track i, and k is the total number of tracks used in an optimal solution.

Step 1: Initialize the priority queue. Add the root node which consists of no net.

Step 2: Pop the priority queue to get the least-cost node X.

Step 3: If node X is a dominated node, then go to Step 2.

Step 4: Let VCG(X) be the vertical constraint graph for the remaining net set $R(X)$. Find the set of all the vertices with no predecessor in the VCG(X). Let this set be S.

Step 5: Construct the horizontal constraint graph of S.

Step 6: Find all the maximal cliques in S. For each maximal clique C, do the following:

Step 6.1: Create a new son node Y for node X, where Y consists of elements in C, and evaluate the cost value of node Y.

Step 6.2: If this newborn node Y has exhausted all of the nets, then stop. The path from the top of the tree to this node represents an optimal solution; otherwise, insert this node into the priority queue.

Step 7: Go to Step 2.

In the previous discussion, we indicated that we had to solve two problems. The first one is to find maximal cliques in a horizontal constraint graph which is a comparability graph [4]. We can extract a Hasse diagram [15] from this graph. Then, finding maximal cliques can be done by conducting a breadth-first searching in this Hasse diagram [17]. The second one is for implementing the dominance

TABLE I
EXPERIMENTAL RESULTS OF DATA FROM [18]

case no.	net no.	column no.	dens.	opt.	generated nodes	dominated nodes	CPU time (sec.)
1 (ex1)	21	43	12	12	61	0	0.830
1* (ex1)	21	43	12	12	616	58	11.370
2 (ex3a)	30	90	15	15	688	0	10.410
3 (ex3c)	54	103	18	18	356	0	8.570
4 (ex4b)	57	119	18	18	1,029	7	23.250
5 (ex5)	64	128	20	20	1,218	0	22.590
6 (ex6)	72	174	19	27	6,849	4,145	329.750

dens.: channel density of this case.
opt. : the minimum number of tracks used in this case.

TABLE II
EXPERIMENTAL RESULTS OF RANDOMLY GENERATED DATA

case no.	net no.	column no.	dens.	opt	generated nodes	dominated nodes	CPU time (sec.)
1	21	126	13	13	1,006	508	23.750
2	48	131	15	15	1,000	0	17.190
3	63	146	14	14	1,969	0	38.600
4	51	129	13	13	2,221	737	90.530
5	56	134	15	15	618	9	16.340
6	51	134	14	14	8,632	750	276.740
7	54	140	13	13	825	0	13.990
8	55	140	14	14	687	1	17.130
9	54	137	14	14	941	0	17.800
10	57	139	27	27	2,394	0	38.740

TABLE III
EXPERIMENTAL RESULTS OF REAL WORLD DATA

case no.	net no.	column no.	dens.	opt.	generated nodes	dominated nodes	CPU time (sec.)
1	48	118	14	14	1,741	0	17.550
2	58	125	12	12	6,039	0	74.080
3	50	107	11	11	3,612	0	37.850
4	60	134	10	10	8,635	0	100.480
5	35	77	12	12	1,132	0	8.410
6	50	112	13	13	3,965	0	38.270
7	39	92	12	12	492	0	5.680
8	44	88	11	11	712	0	7.970
9	41	84	11	11	807	0	7.620
10	36	84	9	9	1,272	0	9.990
11	37	70	9	9	790	0	6.160
12	30	69	7	7	617	0	4.700

rules. Our dominating mechanism requires a special data structure in which some nondominated nodes will be stored, and it can be easily decided, through this data structure, whether a node is dominated or not. In fact, the checking step is $O(N)$, where N is the number of nets [17].

We now list the worst case performance and the average case performance of the Algorithm OCR. Each proof is omitted and can be found in [17]. For a channel specification, let N be the total number of nets, M be the number of columns in the channel, and w be the Dilworth number of the corresponding vertical constraint graph.

Theorem 2: The time complexity of Algorithm OCR which solves the channel routing problem is $O(M \times w \times d \times (d+1)^{w-1})$, where $d = \lceil N/w \rceil$.

Theorem 3: For a bounded channel routing problem whose vertical constraint graph is composed of w disjointed chains and each chain has d nets, $d \times w = N$, the asymptotic value of the average number of nodes generated in the searching tree is $O(N)$.

In a channel routing problem, a case is difficult to solve if the channel specification involves a large number of vertical constraints. Imagine a case where there is no vertical constraint. The vertical constraint graph will consist of N nets not related to one another. That is, there will be N chains with depth 1. In such a situation, this problem can be solved in time complexity $O(N \log N)$. In a difficult case, d is very large and w is bounded. Therefore, Theorem 3 indicates that our algorithm is suitable for solving problems.

V. EXPERIMENTAL RESULTS

Algorithm OCR has been coded in Pascal and run on the VAX 11/750 computer. We did not allow dogleg [2] in our experiments. Examples 1-6 used the data taken from [18]. Table I lists the CPU time, the number of expanded nodes, the number of dominated nodes, and the number of tracks in an optimal solution of those cases. In Table II, we list some results of randomly generated cases. These cases are generated according to the following assumption.

1) The net length is assumed to be a random variable with normal distribution.

2) The number of nets in each case is assumed to be a random variable with normal distribution.

3) The number of columns ranges from 120 to 140.

We also tested our algorithm on many real world cases from the Electronics Research and Service Organization located in Hsinchu, Taiwan. The most difficult case, involving 60 nets, was solved in less than 2 min CPU time. Table III lists some of the results.

The results indicate that the Deutsch difficult case (6 in Table I) can be solved in about 5.5 min. The number of dominated nodes is over half the number of the generated nodes.

In our experience, the auxiliary cost function is an important factor. With this auxiliary cost function, Algorithm OCR may expand fewer nodes. In Table I, Examples 1 and 1* used the same input data. Example 1* did not use the auxiliary cost function, and therefore generated a much greater number of nodes.

VI. Concluding Remarks

According to our experimental results, the minimum number of tracks needed for a channel specification is almost equal to the channel density of that channel specification. Especially, if the length of the longest path in the vertical constraint graph of a case is much smaller than the channel density of that case, then the minimum number of tracks needed for this case is almost equal to the channel density of it. By our experience, some sophisticated heuristic algorithms can solve these cases and their solutions are nearly optimal. As for complex cases, whose number of vertical constraints is very large, the solutions of heuristic algorithms generally are not good. Our algorithm is suitable for these complex cases, and the computation time is reasonable. Therefore, our suggestion is to apply Algorithm OCR to solve the complex cases and apply heuristic algorithms to solve the simple cases.

The cost evaluation function used by Algorithm OCR is admissible [14]. That is, the algorithm will return an optimal solution. Admissible search strategies limit the selection of cost functions and will expand a large amount of unnecessary nodes. Nonadmissible search strategies [14] can avoid these drawbacks and very frequently discover optimal solutions. We can modify our cost evaluation function to be a nonadmissible one. In our experience, it will speed up the computation time, and its solution is near optimal.

References

[1] M. Burstein and R. Pelavin, "Hierarchical channel router," in *Proc. 20th Design Automat. Conf.*, 1983, pp. 591-597.

[2] D. N. Deutsch, "A dogleg channel router," in *Proc. 20th Design Automat. Conf.*, 1976, pp. 425-433.

[3] R. P. Dilworth, "A decomposition theorem for partially ordered sets," *Ann. Math.*, vol. 51, no. 2, pp. 161-166, 1950.

[4] P. C. Gilmore and A. J. Hoffman, "Characterizations of comparability and interval graphs," *Abstract Int. Congress Math.*, Stockholm, 1964, pp. 29-29.

[5] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. Design Automat. Workshop*, 1971, pp. 155-169.

[6] T. Ibaraki, "The power of dominance relations in branch-and-bound algorithms," *J. ACM*, vol. 24, no. 2, pp. 264-279, 1977.

[7] B. W. Kernighan, D. G. Schweikert, and G. Persky, "An optimum channel-routing algorithm for polycell layouts of integrated circuits," in *Proc. 10th Design Automat. Workshop*, 1973, pp. 50-55.

[8] D. E. Knuth, *The Art of Computer Programming, Vol. 3*. Reading, MA: Addison-Wesley, 1973.

[9] A. S. LaPaugh, "Algorithms for integrated circuit layout: Analytic approach," Ph.D. dissertation, MIT Lab Comput. Sci., Nov. 1980.

[10] T. Morin and R. Marsten, "Branch-and-bound strategies for dynamic programming," *Oper. Res.*, vol. 24, pp. 611-627, 1976.

[11] N. Nilsson, *Problem-Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.

[12] N. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980.

[13] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[14] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley Series in Artificial Intelligence, 1984.

[15] F. Preparata and R. Yeh, *Introduction to Discrete Structures: for Computer Science and Engineer*. Reading, MA: Addison-Wesley, 1973.

[16] T. G. Szymanski, "Dogleg channel routing is NP-complete," *IEEE Trans. Comput.-Aided Design*, vol. CAD-4, no. 1, pp. 31-41, 1985.

[17] J. S. Wang, "An efficient algorithm to yield an optimal solution for the channel routing problem in VLSI," Ph.D. dissertation, Instit. of Comput. and Decision Sci., National Tsing Hua University, Hsinchu Taiwan, Republic of China, 1986.

[18] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. Comput.-Aided Design*, vol. CAD-1, no. 1, pp. 25-35, 1982.

Notes on Shuffle/Exchange Type Permutation Sets

SHING-TSAAN HUANG

Abstract—**This paper studies properties of some shuffle/exchange type permutation sets from operational points of view. The permutation sets studied are Ω, Ω^{-1}, Ψ, $\mathfrak{L}$, $\mathfrak{U}$, where Ω and Ω^{-1} are, respectively, the *Omega* and *inverse Omega* permutation sets, $\Psi \equiv (\Omega \cap \Omega^{-1})$, and $\mathfrak{L}$ and $\mathfrak{U}$ are, respectively, the admissible *lower* and *upper* triangular permutation sets. Several intuitive operations are introduced. Based on these operations, important known results relating to these sets are readdressed. The recursive nature of the sets is also discussed.**

Index Terms—**Inverse Omega permutations, lower triangular permutations, Omega permutations, shuffle/exchange network, upper triangular permutations.**

I. Introduction

Considerable efforts have been devoted to the study of *shuffle-exchange* type *permutation* (abbreviated as *pm*) networks. Batcher [1], Stone [10], Lenfant [5], and Nassimi and Sahni [6] have shown that the network can *admit* a wide variety of pms useful in parallel processing. Lawrie has proposed the *Omega* network and studied its admissible pm set Ω [4]. Pease has presented the *indirect binary n-cube* network, which is known to be equivalent to the *inverse Omega* network, and studied its admissible pm set Ω^{-1} [8]; further, in his article, two classes of pm sets $\mathfrak{L}$ and $\mathfrak{U}$, named as *admissible lower triangular* pms and *admissible upper triangular* pms, respectively, were also studied. The equivalence relationship among various pm networks has been studied by Wu and Feng [11], and Parker [7]. Latter, Steinberg has studied the mathematical properties among these pm sets [9]. Recently, Huang and Tripathi have presented an operational model to describe various pm networks [2] and pointed out that a lot of pms mentioned in the literature are in the pm set Ψ [3], where $\Psi \equiv (\Omega \cap \Omega^{-1})$. In this paper, based on the operational model presented in [2], we first present some operations on pm; and then, according to the operations, we readdress important results from the above articles. Finally, we discuss the recursive nature of these pm sets.

A pm $\pi(i)$, $0 \le i \le N-1$, can be represented by an $N \times n$ matrix $[X_n \cdots X_1]$ with row i representing $\pi(i)$ in binary form [2]. In the rest of this paper, we assume that the pm is represented in this matrix form and let τ denote the *identity* pm as shown in Fig. 1. To emphasize n, we may use the superscript (n), for example, $\pi^{(n)}$. Note that a pm remains as a pm after rearranging its columns or rows. We also assume that $\pi = [X_n \cdots X_1]$ and $\tau = [Z_n \cdots Z_1]$ unless otherwise stated.

We call an operation that replaces the ith column in a pm with another vector *i-exchange* operation. In Section II, we define some classes of i-exchange operations and then define the pm sets according to these operations. Known important results about the pm sets are readdressed in Section III. Then, in Section IV, we present some recursive properties of these pm sets. Section V gives some examples to show the potential power of our operational approach. Finally, conclusions are drawn in Section VI.

II. i-Exchange Operations

The first class of i-exchange operations is *i-Cyclic-shifting*, denoted as C_i. An instance of C_i on π can be $C_i(\pi, c)$ which cyclically shifts (or vertically rotates) the ith column of π, viz., X_i, c bit po-

Manuscript received September 8, 1987; revised July 3, 1989. This work was supported by the National Science Council of the Republic of China under Contracts NSC76-0408-E007-07 and NSC78-0408-E007-08.

The author is with the Institute of Computer Science, National Tsing Hua University, Taiwan, Republic of China.

IEEE Log Number 9034547.