Journal of Physical Sciences, Vol. 11, 2007, 199-210

A Mimetic Algorithm for Computing a Nontrivial Lower Bound on Number of Tracks in Two-Layer Channel Routing

Rajat K. Pal, Debasri Saha, and Samar Sen Sarma

Department of Computer Science and Engineering
University of Calcutta
92, A. P. C. Road, Kolkata – 700 009, India

Received November 27, 2007; accepted December 10, 2007

ABSTRACT

Study of algorithms and its design can be progressed in various dimensions. In this paper, we have a definite refinement of lower bound on the number of tracks required to route a channel. The attack is from a complementary viewpoint. Our algorithm succeeds to avoid all kinds of approximation. The approach performs exact mapping of the problem into graphical presentation and analyzes the graph taking help of mimetic algorithm, which uses combination of sequential and GA based vertex coloring. Performance of the algorithm depends on how effectively mimetic approach can be applied selecting appropriate values for the parameters to evaluate the graphical presentation of the problem. This viewpoint has immense contribution against sticking at local minima for this optimization problem. The finer result clearly exemplifies instances, which give better or at least the same lower bound in VLSI channel routing problem.

Key words : *Manhattan routing model, Channel routing problem, Constraint graphs, Maximum independent set, Mimetic algorithm.*

1. Introduction

1.1. Channel Routing Problem

The channel routing problem (CRP) of area minimization is NP-hard in nature [6]. Extensive effort and attention has been attempted to tackle it. With the advancement of VLSI technology, as millions of gates have been accommodated in a tiny chip area, wiring the terminals of logic blocks altogether using minimum possible area has become a tedious task. If electrically equivalent pins are wired using rectangular routing region with terminals only on opposite sides, this strategy is termed as channel routing.

CRP is a constrained optimization problem, where horizontal span of nets are assigned to horizontal tracks, avoiding conflicts so that track requirement is minimized. As CRP is NP-hard [3, 7, 8], to design an algorithm with much lower complexity, we have taken heuristic support. As practical lower bound deviates much from the trivial one, our algorithm focuses on the computation of nontrivial lower bound on the number of tracks. The evolutionary techniques of mimetic algorithm, which efficiently handles hybrid optimization problems, are effectively incorporated here to find a better nontrivial solution. It generates near-optimal results for a number of well-known benchmark channels in reasonable time.

Here we consider grid based reserved layer Manhattan routing model, which is rectilinear in nature and each layer is restricted to accommodate a certain type (horizontal or vertical) of wire segments.

1.2. Constraints of CRP and their Significance

Routing of wires should satisfy both kind of constraints, horizontal constraints and vertical constraints. Two nets n_i and n_j are said to have horizontal constraints, if their horizontal spans have at least one column common. Two nets n_i and n_j are said to have vertical constraints, if there exists a column such that the terminal on the top of the column belongs to net n_i and the terminal at the bottom of the column belongs to net n_j, or vice versa.

These constraints can be well visualized by two constraint graphs, horizontal constraint graph (HCG) and vertical constraint graph (VCG) [6, 10].

HCG, G=(V,E) is an undirected graph where each vertex $v_i \in V$ represents a net n_i and each edge $(v_i,v_j) \in E$ represents horizontal constraint between net n_i and net n_j. It signifies that if there is an edge between vertices v_i and v_j, then nets n_i and n_j cannot be placed in the same track.

Horizontal constraint can have a complementary representation through horizontal non-constraint graph (HNCG). HNCG, G=(V,E′) is an undirected graph where each vertex $v_i \in V$ represents a net n_i and each edge $(v_i,v_j) \in E'$ indicates that net n_i and net n_j are horizontal constraint-free, i.e., horizontal spans of nets n_i and n_j have no common column. It implies that if there is an edge between vertices v_i and v_j, then nets n_i and n_j can be placed in the same track if only horizontal constraint is taken into account.

VCG, G=(V,A) is a directed graph where each vertex $v_i \in V$ represents a net n_i and each directed edge $\langle v_i,v_j \rangle \in A$ represents vertical constraint between net n_i and net n_j such that there exists a column for which the top terminal belongs to net n_i and the bottom terminal belongs to net n_j. Interpretation of VCG is that if there is a directed edge from vertices v_i to v_j, then net n_i must be placed in a track above the track where net n_j is placed. That means it emphasizes the ordering of net assignments in the channel.

The maximum number of nets that cross a column gives the knowledge of channel density, d_{max}. If we neglect vertical constraint, minimum number of track requirement is equal to d_{max}. This information is extracted either from HCG or from HNCG. In case of HCG, computation of clique number generates the value of channel density whereas if HNCG is considered, we have to calculate independence number. Here we introduce the definitions of clique number and independence number of a graph.

Definition 1: *Clique number* of a graph is the size of maximum complete subgraph of the graph.

Definition 2: A set of vertices in a graph is said to be an independent set of vertices or simply *independent set* if no two vertices in the set are adjacent.

Definition 3: A *maximal independent set* is an independent set to which no other vertex can be added without destroying its independence property.

Definition 4: The number of vertices in the largest independent set of a graph is called the *independence number* of the graph.

On the other hand, VCG contributes the value of v_{max}, which is nothing but the length of longest chain in VCG. It indicates that, if we consider only vertical constraints, at least v_{max}-number of tracks is required.

The rest of the paper is organized as follows. Section 2 discusses the motivation of the work. Section 3 discusses the proposed algorithm and Section 4 throws light on the time complexity of the algorithm. Section 5 illustrates the execution of the algorithm by an example. Section 6 focuses on the definite refinement on minimum number of tracks to route a channel and discusses the empirical observations on some randomly generated instances. Section 7 extends our proposed algorithm for two-layer restricted dogleg routing model. Section 8 concludes the paper and discusses scope for future work.

2. Motivation of the Work

Our work is motivated, as we have analyzed a lot of practical instances of channel, which cannot be routed using either d_{max}- or v_{max}-number of tracks. Apparently $\max(d_{max}, v_{max})$ is formulated as an estimate of *trivial lower bound*. But simultaneous consideration of both the constraints generates a practical situation where a greater number of tracks are necessary to route a channel. It encourages us to combine the information from two constraint graphs into a single one, so that the resulted composite constraint graph can conjointly helps us to find a *nontrivial lower bound*.

HCG is an interval graph, whereas its complement graph is a comparability graph [2]. The common feature of them is that they are both perfect in nature. A graph is said to be perfect, if it has no induced subgraph with odd cycle of length greater than or equal to five.

But VCG can be any directed acyclic graph (if we take a channel only with cycle-free VCG). If we proceed by extracting constraint based information from VCG and incorporating those into HNCG, it results into a modified HNCG, which may not still remain perfect in nature. Although clique number or independence number of perfect graph is polynomial time computable, the possibility for modified HNCG of being non-perfect restricts us guaranteeing a deterministic polynomial time algorithm for independence number computation.

Success of mimetic algorithm in handling NP-hard optimization problems inspired us to introduce it in our problem solving [4, 5]. In our paper, mimetic algorithm optimally colors the vertices of the composite graph. The result is equivalent to finding a maximal independent set of maximum cardinality.

In paper [9], it is deliberately kept the composite constraint graph (modified HCG) chordal, as clique number of chordal graph is polynomial time computable. But to do so, some vertical constraint based information is lost, which is treated as approximation. Hence there the modified HCG reflects only approximated lower bound, not the exact one. Here we preserve all constraint related information in modified HNCG and this information is processed using GA operators to produce practical lower bound.

3. The Proposed Algorithm

3.1. Construction of Composite Graph

We propose a hybrid GA based heuristic algorithm to determine a nontrivial lower bound on number of tracks required to route a channel in polynomial time. An edge between vertices v_i and v_j in HNCG signifies that, nets n_i and n_j have no horizontal overlapping. That does not mean the nets can be placed in the same track, as vertical constraint may impose ordering on their tracks. In VCG, if directed edges $<v_i,v_j>$ and $<v_j,v_k>$ are present, that indicates net n_i has to be placed above net n_j and net n_j above net n_k. Hence net n_i has to be placed above net n_k. This transitive closure property is strictly followed by vertical constraints. So net n_i cannot be accommodated with net n_k, even if those are horizontal constraint-free. It is focused that none of the constraint graphs can alone cover all constraint information. So we extract this vertical constraint based information from VCG and incorporate those into HNCG to highlight all constraint information through a single graph.

We find out all possible directed paths between each pair of source (indegree zero) and sink (outdegree zero) vertices in VCG, then apply transitive closure property (if $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$) to construct an edge list E, which contains edges between all pairs of vertices having a directed path between them in VCG, but without only the (directed) edges between them. The directed edges already present in VCG reflect direct vertical constraints; hence those are automatically covered by horizontal constraint consideration. Hence E contains only those path information that reflect indirect or derived vertical constraints.

An edge (v_i,v_j) of edge list E, if present in HNCG, indicates that the corresponding nets n_i and n_j are not horizontally constrained but only vertically. Our strategy is to delete all such edges from HNCG. The modified HNCG, thus obtained, is termed as *composite graph* as it focuses combined effect of all constraints.

Definition 5: For *composite graph* G=(V,E), each vertex $v_i \in V$ represents a net n_i and each edge $(v_i,v_j) \in E$ implies that the corresponding nets n_i and n_j are constraint-free and can be placed in same track.

Conversely we can say, two disconnected vertices v_i and v_j reflect the fact that corresponding nets n_i and n_j are mutually constrained, hence occupy separate tracks. Independence number I, i.e., the maximum number of mutually disconnected vertices of the composite graph gives an estimate of lower bound (Lbound) on the number of tracks required to route a channel.

3.2. Computation of Independence Number using Mimetic Algorithm

The problem of finding independence number I of a composite graph is mapped into the problem of proper coloring of vertices, where connected vertices are colored with distinct colors. Our algorithm proceeds with proper coloring of composite graph satisfying the objective that as many vertices as possible are colored by each color applied. That means, if each color is assigned to as many vertices as possible obeying proper coloring, the maximum number of vertices colored with identical color specifies independence number I.

In this context, the order of sequential coloring of vertices is of great significance. The vertices of composite graph are arranged in increasing order of their degree and considered for proper coloring in this sequence. We stack for use as many colors as the number of vertices in composite graph. Each color is encoded as an integer. GA works by evolving a population of strings over generations. We use random selection of a color, consider vertices in minimum degree sequence, and continue assigning the color till the violation of proper coloring, followed by selection of another color. Fitness value of a string is evaluated as the maximum occurrences of a single color (integer) in the string. GA attempts to optimize this fitness function through effective application of GA parameters *reproduction*, *crossover* and *mutation* [1] with appropriate probability. Reproduction emphasizes survival of highly fit strings. Crossover provides encouraging results against sticking to local optima. Random selection of mutation location also helps to reach global minima.

3.3. Detection of Obstruction Condition

Let us consider the following two channel specifications.

TOP:	3	1	2	0	2	0		TOP:	1	1	4	0	2	0
BOTTOM:	0	3	0	1	4	4		BOTTOM:	0	3	0	3	4	2

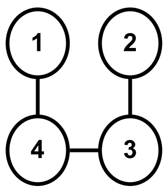
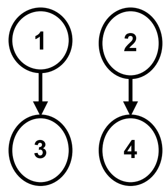
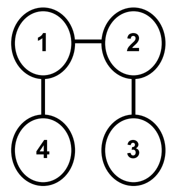
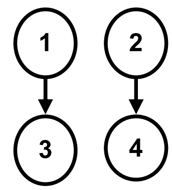





Figure 1: The HNCG. **Figure 2:** The VCG. **Figure 3:** The HNCG. **Figure 4:** The VCG.

In both cases, composite graph is same as HNCG. Independence number I is 2 but track requirement is 3, as net 1 and net 2 cannot be placed in same track for the first channel, and net 3 and net 4 cannot be placed in same track for the second channel. So, at least three tracks are required to route each of the channels.

Lemma 1: For a pair of directed paths (chain) from source to sink vertices, with length difference less than or equal to one and at least one with length v_{max}, if source vertices, or sink vertices, or both pairs are disconnected in HNCG, then at least one extra track is essentially required to route the channel.

Our proposed algorithm searches for the presence of *obstruction condition* (as stated in Lemma 1), and if found, at least one extra track is needed. Hence *minimum increment in number of track requirement*, *INCR* is 1.

3.4. Algorithms

Algorithm MIMETIC_LBOUND

Input: Channel specification.

Output: Lbound, a nontrivial lower bound on the number of tracks.

Step 1: Construct HNCG and VCG from channel specification.

Step 2: Using the transitive closure property, compute the list of edges, E between all possible pairs of vertices having shortest directed path length two or more between them in VCG.

Step 3: If E is empty, consider HNCG as composite graph (or modified HNCG). Go to Step 5.

Step 4: Delete each edge $e \in E$ from HNCG, if e is present in HNCG. Finally resulted graph is denoted as composite graph (or modified HNCG).

Step 5: If the composite graph does not contain any edge (having only isolated vertices), then Lbound is same as the number of vertices in composite graph, else compute independence number, I of the composite graph using mimetic algorithm.

Step 6: Check for the presence of *obstruction condition*.
If present, compute increment in lower bound, INCR due to that, else INCR = 0.
Finally, Lbound = I + INCR.

Following are the steps of mimetic algorithm to compute the independence number of a graph.

Mimetic Algorithm I_number

Input: Composite graph, size of initial population, number of iteration n, crossover probability pcross, mutation probability pmutate.

Output: I, the independence number of the composite graph.

Step 1: Generate initial population containing valid and unique strings of colors using sequential vertex coloring.

Step 2: Compute maximum fitness value, max_fitness, of strings in current population.
Repeat up to Step 6 for n times.

Step 3: Select strings of high fitness value to generate population for crossover (reproduction). Repeat Step 4 for ncross[#] times.

Step 4: Select parents and crossover site; perform crossover.
Check validity of new strings; if valid, replace previous one by it.
Repeat Step 5 for nmutate[#] times.

Step 5: Select string for mutation, site and replacing color; perform mutation.
Check validity of new strings; if valid, replace previous one by it.

Step 6: Compute maximum fitness value, new_max_fitness, of the new generation population.
If new_max_fitness > max_fitness, max_fitness ← new_max_fitness; replace current population with new generation population.

Step 7: I ← max_fitness.

[#] (Compute ncross (number of crossover) from pcross and nmutate (number of mutation) from pmutate.)

4. Complexity Analysis of MIMETIC_LBOUND

Complexity calculation in mimetic algorithm based design is not straightforward. This paper emphasizes on finding a better nontrivial lower bound than the earlier deterministic algorithm [9]. Let us try to give some highlights of time complexity of our algorithm. Sequential vertex coloring requires $O(n^2)$ time, where n is the number of nets. The initial population of genetic algorithm is thus obtained in $O(n^2)$ time complexity.

For mimetic algorithm based heuristic search, we know that it is suitable for

MIMD parallel computing and distributed computing environment as these are composed by network of workstations. However, we have seen that CPU time required for executing our algorithm using single Pentium4 processor is reasonable for all practical purposes.

5. Illustration with an Example

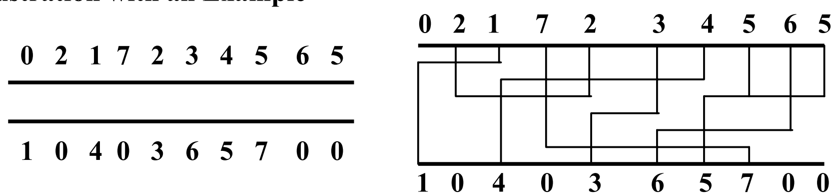


Figure 5: A channel instance and its routing solution.

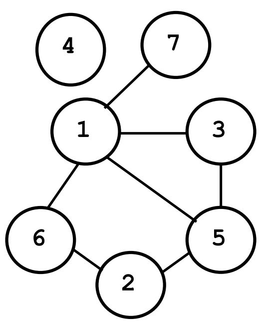


Figure 6: The HNCG.

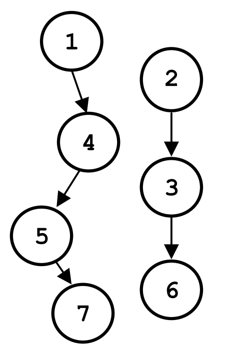


Figure 7: The VCG.

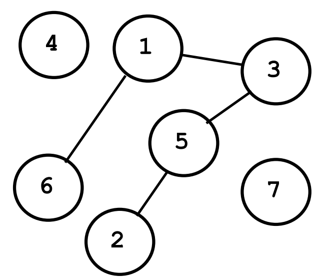


Figure 8: The composite graph.

Using transitive closure property, the final edge list E is constructed; E = {(1,5), (1,7), (4,7), (2,6)}. Edges in this list indicate derived or indirect vertical constraints between corresponding nets. Among these edges (1,5), (1,7), and (2,6) are present in HNCG, and those have to be eliminated from HNCG. Deletion of those edges generates composite graph. The maximum independent set is {2, 3, 4, 6, 7}. That is, independence number, I = 5. Analyzing VCG, it is revealed that there are two directed paths from source to sink vertices of lengths 4 (= v_{max}) and 3. The paths are ($1 \rightarrow 4 \rightarrow 5 \rightarrow 7$) and ($2 \rightarrow 3 \rightarrow 6$). The source vertices 1 and 2 are horizontally constrained, and the edge (1,2) is absent in HNCG. Thus *obstruction condition* is satisfied for this channel instance. So, INCR = 1. Hence minimum number of track requirement by our algorithm MIMETIC_LBOUND is 5+1 or 6. Practical solution shows that, the minimum number of tracks requirement is also 6. Hence the result obtained by MIMETIC_LBOUND tallies with practical solution.

6. Refinement of Lower Bound using Our Algorithm

Theorem 1: MIMETIC_LBOUND computes exact lower bound on the number of track requirement to route a channel, and result is better or at least equal to that found using LOWER_BOUND algorithm [9].

We demonstrate the refinement in results achieved by MIMETIC_LBOUND in comparison to other algorithms, in Table 1 below.

Table 1: Lower bound using MIMETIC_LBOUND and comparison with other algorithms.

Channel instance	d_{max}	v_{max}	$max(d_{max}, v_{max})$	Lbound by our algorithm	CPU time	Best solution known [9]
CH1	4	4	4	6	0.002s	5
CH2	3	5	5	6	0.0023s	5
CH3	4	4	4	6	0.0025s	5
CH4	4	4	4	6	0.0024s	5
CH5	5	5	5	7	0.0034s	6
RKPC1	3	3	3	4	0.002s	4
RKPC6	4	5	5	7	0.11s	7
RKPC8	5	5	5	7	0.06s	7
RKPC9	6	6	6	10	0.16s	10
DDE	19	23	23	28	1min 54.16s	28

Table 2: Suitable values of GA parameters to obtain optimum solutions for some channel instances using MIMETIC_LBOUND.

Channel instance	GA related parameters for optimum Lbound			
	Initial population	Number of iteration	Crossover probability	Mutation probability
CH1	10	2	.4	.001
CH2	10	2	.4	.001
CH3	10	2	.4	.001
CH4	10	2	.4	.001
CH5	20	6	.8	.001
RKPC1	16	6	.8	.001
RKPC6	14	4	.6	.001
RKPC8	12	2	.4	.001
RKPC9	30	4	.8	.001
DDE	140	12	.8	.001

The result is achieved implementing MIMETIC_LBOUND in MATLAB using Pentium4 machine with clock frequency 1.5 GHz. Channel instances *CH1* through *CH5* (see Appendix) clearly demonstrate refinement in results. For next four channel instances *RKPC1*, *RKPC6*, *RKPC8*, and *RKPC9* [6], MIMETIC_LBOUND results tally with previous results. MIMETIC_LBOUND also provides result as good as earlier computed lower bound for *Deutsch's difficult example (DDE)* [9]. Column *CPU time* shows that the time required in computing the number of tracks necessary to route a channel is negligible for most of the instances; even for *DDE* it is not very large.

Table 2 shows some GA related parameters in order to compute optimum Lbound. Size of initial population is increased, in general, with the number of nets of the channel instances. For the channel instances, for which computation of independence number of the corresponding composite graphs has greater probability of sticking at local maxima, higher value for both crossover probability and number of iterations help us to achieve optimum solution.

7. Two-Layer Restricted Dogleg Routing

For channels with multi-terminal nets, restricted doglegging often removes cycles from VCG and can route such channels. It sometimes produces better routing solution in terms of channel area or number of tracks required. Our algorithm can invariantly be applied for multi-terminal nets, if horizontal wire segment of such net is split into set of two-terminal subnets and HCG (or HNCG) and VCG are constructed as follows.

For both HCG, $G_{HC} = (V,E_{HC})$ and VCG, $G_{VC} = (V,E_{VC})$, V is the set of vertices corresponding to two-terminal subnets of nets. If e_{pi} and e_{qj} are two subnets of two different nets n_i and n_j, respectively, then $(e_{pi},e_{qj}) \in E_{HC}$, when e_{pi} and e_{qj} overlap. HNCG, G_{HNC} is obtained by complementing G_{HC}. For constructing edges of VCG, if nets n_i (with a terminal on the top) and n_j (with a terminal at the bottom) both cross through some column c, where l_i and r_i are subnets of net n_i, and l_j and r_j are subnets of net n_j to the left and right of column c, then directed edges $<l_i,l_j>$, $<l_i,r_j>$, $<r_i,l_j>$, and $<r_i,r_j>$ have to be introduced in VCG. Construction of HNCG and VCG, and hence lower bound on number of tracks for channels with multi-terminal nets can be demonstrated with the help of an example, as shown below.

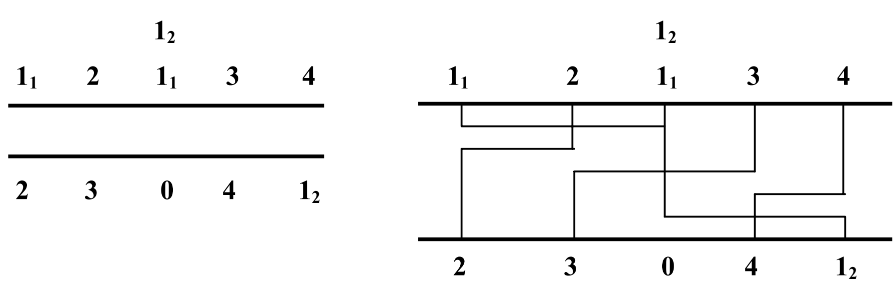
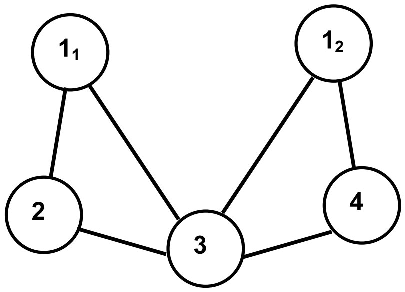
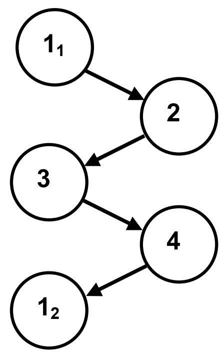


Figure 9: A channel instance and its dogleg routing solution.

Here the VCG of the example channel forms a cycle. So we dogleg net 1 and obtain two subnets 1_1 and 1_2 for their assignment to different tracks, as shown in Figure 9.





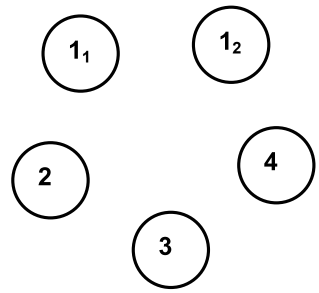


Figure 10: The HNCG, G_{HNC}. **Figure 11:** The VCG, G_{VC}. **Figure 12:** The composite graph.

The HNCG and the VCG based on two-terminal subnets of the nets belonging to this example channel are shown in Figures 10 and 11, respectively. As a result, the composite graph for this channel instance is a graph that contains only five isolated vertices; see Figure 12. Hence, the independence number I is same as 5, and the lower bound on number of track requirement is also 5 that tallies with the practical solution for routing the nets, as shown in Figure 9.

8. Conclusion

Heuristic algorithms, in general, outperform approximation algorithms. In this paper, we have developed an algorithm to compute a nontrivial lower bound on the number of tracks required to route a two-layer (VH) channel. The algorithm presented is GA based and exhaustive in nature as the problem of computing minimum area routing solution is NP-hard. The deterministic version of computing a nontrivial lower bound is presented in [9], that took time $O(n^4)$ for a channel of n nets. Here we have considered the exact problem of computation of lower bound and solved the problem by a mimetic algorithm that computes almost optimal number of tracks required for most of the practical channel instances under consideration. The result is encouraging, as it shows better lower bounds on number of tracks in many instances but never worse.

The extension of the work in multi-layer environment is our next projected extension of the work.

Appendix :

CH1:	TOP:	0	2	1	7	2	3	4	5	6	5
	BOTTOM:	1	0	4	0	3	6	5	7	0	0

CH2:	TOP:	9	8	7	5	6	1	0	0	2	4	3
	BOTTOM:	0	0	9	8	7	6	2	1	4	3	5

CH3:	TOP:	0	2	1	9	2	4	0	3	5	6	7	0
	BOTTOM:	1	0	3	0	4	6	9	5	7	8	0	8

CH4:	TOP:	0	4	1	4	2	0	3	5	9	6	7	0
	BOTTOM:	1	0	3	2	6	9	5	7	0	8	0	8

CH5:	TOP:	0	4	1	11	4	2	3	5	6	7	8	0	9	0
	BOTTOM:	1	0	3	0	2	6	5	7	8	9	10	11	0	10

REFERENCES

[1] Deb K., *Optimization for Engineering Design*, Prentice-Hall of India, 1995.

[2] Golumbic M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[3] LaPaugh A. S., *Algorithms for Integrated Circuit Layout: An Analytic Approach*, Ph.D. thesis, Lab. for Computer Sc., MIT, Cambridge, 1980.

[4] Mazumder P. and E. M. Rudnick, *Genetic Algorithms for VLSI Design, Layout and Test Automation*, Prentice-Hall of India, 1999.

[5] Merz P., On the Performance of Mimetic Algorithms in Combinational Optimizations, Second Workshop on Mimetic Algorithms, San Francisco, USA, 2001.

[6] Pal R. K., *Multi-Layer Channel Routing: Complexity and Algorithms*, Narosa Publishing House, New Delhi (Also published from CRC Press, Boca Raton, USA and Alpha Science International Ltd., UK), 2000.

[7] Pal R. K., S. P. Pal, A. K. Datta, and A. Pal, NP-Completeness of Multi-Layer No-Dogleg Channel Routing and an Efficient Heuristic, Proc. of Sixth VSI/IEEE International Conference on VLSI Design, Bombay, India, pp. 80-83, 1993.

[8] Pal R. K., S. P. Pal, and A. Pal, On the Computational Complexity of Multi-Layer Channel Routing, Technical Report No.: TR/IIT/CSE/92/02, Dept. of Computer Sc. & Engg., I.I.T., Kharagpur, 1992.

[9] Pal R. K., S. P. Pal, and A. Pal, An Algorithm for Finding a Non-Trivial Lower Bound for Channel Routing, INTEGRATION: *the VLSI Journal*, Vol. 25, No. 1, pp. 71-84, 1998.

[10] Schaper G. A., *Multi-Layer Channel Routing*, Ph.D. thesis, Dept. of Computer Sc., Univ. of Central Florida, Orlando, 1989.