2001

Knowledge-based genetic algorithm for layer assignment

Maolin Tang
Queensland University of Technology

Kamran Eshraghian
Edith Cowan University

Daryoush Habibi
Edith Cowan University

10.1109/ACSC.2001.906641
This conference paper was originally published as: Tang, M., Eshraghian, K. , & Habibi, D. (2001). Knowledge-based genetic algorithm for layer assignment. Proceedings of 2001 Australian Computer Science Conference (pp. 184-190). Gold Coast, QLD. IEEE. Original article available here

Knowledge-based Genetic Algorithm for Layer Assignment

Maolin Tang
School of Computing Science and Software Engineering
Queensland University of Technology
QLD 4001, Australia
m.tang@qut.edu.au

Kamran Eshraghian, Daryoush Habibi
School of Engineering and Mathematics
Edith Cowan University
WA 6027, Australia
{k.eshraghian, y.habibi}@ecu.edu.au

Abstract

Layer assignment is an important post-layout optimization technique in Very Large Scale Integrated-circuit (VLSI) layout automation. It re-assigns wire segments in a routing solution to appropriate layers to achieve certain optimization objectives. This paper focuses on investigating the layer assignment problem with application to via minimization, which is known to be NP-complete. In this paper a knowledge-based genetic algorithm for the layer assignment problem is proposed, with the aim of utilizing domain-specific knowledge to speedup the process of evolution and to improve the quality of solutions. Experimental results show that this knowledge-based genetic algorithm can consistently produce the same or better results than a heuristic algorithm and a traditional genetic algorithm.

1. Introduction

Layer assignment is an important post-layout optimization technique in VLSI layout automation. It re-assigns wire segments in a routing solution to appropriate layers to achieve certain optimization objectives. Layer assignment has become an interesting topic since it preserves the wire lengths and topologies of the initial routing solution during optimization and provides considerable flexibility for optimizations of vias, crosstalk, and delays. This paper focus on investigating layer assignment problem with application to via minimization.

Via is a mechanism (hole) for connecting wire segments of a net distributed on different layers in two-layer or multi-layer VLSI routing. However, since via has an associated resistance that affects circuit performance, it is desirable to minimize the number of vias in a VLSI routing.

This layer assignment problem can be stated as follows: Given a collection of nets, each of which consisting of wire segments that electrically connect a set of terminals, find a layer assignment to all the wire segments such that the number of vias required is minimized, any two wire segments in different nets that cross or overlap each other are assigned to different layers, and all the wire segments in the same net are interconnected.

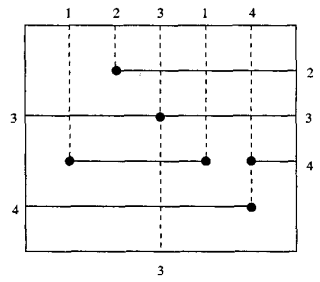
Figure 1 shows an instance of the layer assignment, i.e. Figure 1(a) is an initial routing solution and Figure 1(b) gives the solution after the layer assignment. In this figure, dotted lines represent wire segments on one routing layer, solid lines stand for wire segments on another routing layer, and dots are vias.

From the computational point of view, the layer assignment problem is a NP-complete optimization problem [10]. Hence, a genetic algorithm would be an effective and efficient method for solving the layer assignment problem. In this paper, a knowledge-based genetic algorithm for the layer assignment problem is proposed, with the aim of utilizing domain-specific knowledge to speedup the process of evolution and to improve the quality of solutions.

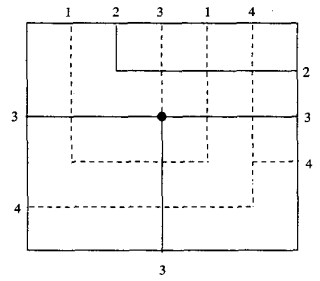
The layer assignment problem can be modeled in *switching graph model* [13]. Under the switching graph model the layer assignment problem is transformed into a so-called *switching graph problem*. This knowledge-based genetic algorithm is based on the switching graph model. For the sake of presentation, we call this knowledge-based genetic algorithm KBGA in this paper.

This paper is organized as follows: firstly, we review the

1530-0900/01 $10.00 © 2001 IEEE



(a)



(b)

Figure 1. An instance of layer assignment

switching graph model. Then, the KBGA is outlined, followed by the representation and the fitness function used in the KBGA. Next, we discuss the genetic operators with focuses on discussing knowledge-based issues. After that, implementation issues are discussed and experimental results are given. Finally, we conclude this knowledge-based genetic algorithm.

2. Review of switching graph model

A feasible layer assignment R can be represented as a LAP (Layer Assignment Problem) graph $G = (V, E)$ [13]. G is a directed bigraph [6] whose vertices set V can be partitioned into two disjointed sets $V_{cluster}$ and V_{via}. The vertices belonging to $V_{cluster}$ are *cluster vertices* and the vertices belonging to V_{via} are *via candidate vertices*. E is the set of directed edges, each of which associates a cluster vertex with a via candidate vertex. Figure 2 is an instance of LAP graph. In the LAP graph, $V_{cluster} = \{c_1, c_2, \cdots, c_7\}$, $V_{via} = \{v_1, v_2, \cdots, v_7\}$.

The LAP graph reflects mutual constraints among via candidate vertices and cluster vertices and possesses the following useful properties:

Property 1. The number of the via candidate vertices whose in-degree and out-degree are both non-zero in a LAP graph equals the number of the vias introduced in the corresponding layer assignment.

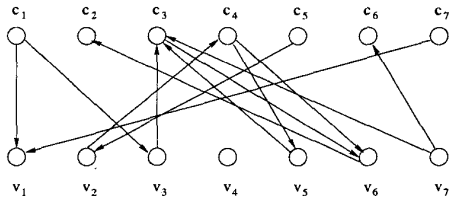


Figure 2. An instance of LAP graph

The vertices whose in-degree and out-degree are both non-zero are *via vertices* and the vertices whose in-degree or out-degree is zero are *non-via vertices*. Denote the in-degree and out-degree of a via candidate vertex v as $id(v)$ and $od(v)$ respectively. Then a LAP graph has the property described below:

Property 2. For $\forall v \in V_{via}$, $id(v) + od(v) \leq 4$.

A *switching graph* $S_G(C)$ of a LAP graph $G = (V, E)$ is defined as a bigraph obtained by reversing the direction of the arcs incident to the cluster vertices in the cluster subset C. $S_G(C) = (V_G, E_G)$ is formally defined as below:

$V_G = V$;
$E_G = E - \{< u_1, u_2 > | (< u_1, u_2 > \in E).and.((u_1 \in C).or.(u_2 \in C))\} + \{< u_2, u_1 > | (< u_1, u_2 > \in E).and.((u_1 \in C).or.(u_2 \in C))\}$.

For example, switching graph $S_G(\{c_3, c_6\})$ of the LAP graph shown in Figure 2 is displayed in Figure 3.

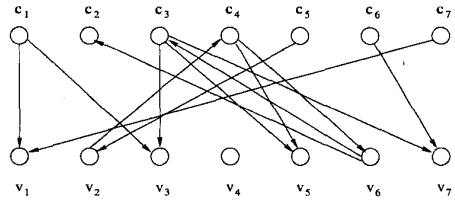


Figure 3. Switching graph $S_G(\{c_3, c_6\})$

Property 3. Assume that l is a feasible layer assignment, G is the LAP graph of l, S is the set of the switching graphs of G, and L is the set of all feasible layer assignments. Then, S and L are one-to-one correspondence.

The following are three corollaries resulting from Property 3.

Corollary 1. The layer assignment corresponding to $S_G(C)$ is a feasible one, where $C \subseteq V_{cluster}$.

Corollary 2. There are 2^n different feasible layer assignments.

Corollary 3. The layer assignment with minimal number of vias corresponds to the switching graph with minimal number of via vertices.

Given a LAP graph G, find a switching graph with minimal number of via vertices. This is so-called *switching graph problem.* The rest of this paper focuses on solving the switching problem problem.

3. Outline of the KBGA

The KBGA operates on a population of individuals, each of which represents a switching graph. Initial population is created by randomly generating a collection of individuals. Each individual i is evaluated using a fitness function $f(i)$. Basically, the KBGA discovers better individuals by allowing the individuals to evolve from generation to generation, and the evolution is realized through reproduction.

The process of reproduction is the point at which evolution takes place. It is implemented by two genetic operations in the KBGA: *crossover* and *mutation.* Crossover is a knowledge-based recombination operator which mixes the genes from two parents to reproduce two offspring. Mutation is another recombination operator to randomly change an allele of an individual for keeping diversity in a population. In order to reproduce better offspring, roulette selection strategy [7] is adopted for selecting parents for reproduction. This selection strategy makes sure that the fitter individuals have more chances to be selected for reproducing fitter offspring in the next generation.

In each generation, the KBGA calculates fitness for all individuals and retains the fittest one. The fittest one then is further evolved by using a hill-climbing operator. The fittest evolved is then compared with the fittest evolved in the previous generation, and the fitter one is kept as the fittest individual in history. Hence, when the KBGA terminates, the fittest individual in history is considered as the optimal switching graph. the KBGA is described in Algorithm 1.

Algorithm 1: Genetic Algorithm for Switching Graph Problem

create initial population P_{old};
for $\forall i \in P_{old}, Cal\_Fitness(i)$;
$i_{best} = Best\_Individual(P_{old})$;
$i_{best} = Hill\_Climbing(i_{best})$;
for generation $= 1$ *to* $MaxGen$ *do*
begin
  for $i = 1$ *to* $\lfloor PopSz/2 \rfloor$ *do*
  begin
    $P_{new} = \phi$;
    $p_\alpha = Select(P_{old})$;
    $p_\beta = Select(P_{old})$;
    $Crossover(p_\alpha, p_\beta, o_\alpha, o_\beta, p_{crossover})$;
    $Mutate(o_\alpha, p_{mutation})$;
    $Mutate(o_\beta, p_{mutation})$;
  end
  $P_{old} = P_{new}$;
  for $\forall i \in P_{new}, Cal\_Fitness(i)$;
  $t_{best} = Best\_Individual(P_{new})$;
  $t_{best} = Hill\_Climbing(t_{best})$;
  if $Cal\_Fitness(i_{best}) < Cal\_Fitness(t_{best})$ *then*
    $i_{best} = t_{best}$;
end
output i_{best}.

In Algorithm 1, P_{old} and P_{new} are the old generation and the new generation respectively, i_{best} retains the fittest individual in the history, $Cal\_Fitness(i)$ is a procedure of calculating the fitness of an individual i, $Best\_Individual(P)$ finds the fittest individual in the population P, $Select(P)$ is a genetic operator which is used for selecting an individual among the population P in the roulette selection strategy, $Crossover(p_\alpha, p_\beta, o_\alpha, o_\beta, p_{crossover})$ is a knowledge-based recombination operator to reproduce two offspring o_α and o_β from their parents p_α and p_β with the probability $p_{crossover}$, $Mutate(o_\beta, p_{mutation})$ is another recombination operator used for mutation with the probability $P_{mutatation}$, $Hill\_Climbing(i)$ is a knowledge-based operator which produces the local optimum from the individual i. $MaxGen$ and $PopSz$ represent the maximal number of generations and population size, respectively.

4. Representation

A switching graph represents a feasible layer assignment. Thus, a chromosome in this genetic algorithm corresponds to a switching graph. Suppose that $G = (V, E)$ is a LAP graph and $S_G(C)$ is a switching graph of G, where $V = V_{cluster} \cup V_{via}$, and $C \subseteq V_{cluster}$. Let $n = |V_{cluster}|$ and $p = |V_{via}|$. Then, $S_G(C)$ is represented as a binary string of n bits,

$$b_1 b_2 \cdots b_n,$$

where,

$$b_i = \begin{cases} 1 & \text{if } c_i \in C \\ 0 & \text{if } c_i \notin C \end{cases}$$

Under this representation, the LAP graph G (a special switching graph with $C = \phi$) is always encoded as $00\cdots0$. For example, the LAP graph shown in Figure 2 is represented as 0000000. The switching graph $S_G(\{c_3, c_6\})$ of the LAP graph shown in Figure 3 is represented as 0010010.

5. Fitness function

The objective of the KBGA is to find a switching graph of a given LAP with minimum number of via candidate vertices. The fewer via candidate vertices a switching graph has, the fitter the corresponding chromosome is. Thus, the fitness of a chromosome i is defined in Equation 1:

$$f(i) = p - via(i) \tag{1}$$

where p is the number of via candidate vertices and $via(i)$ is the number of via vertices in the corresponding switching graph of i. The fitness of a chromosome i is calculated in three steps:

1. Decode the chromosome i to obtain its corresponding switching graph M;
2. Calculate $via(i)$ from M;
3. Calculate $f(i)$ by Equation 1.

In order to decode a chromosome i to obtain its corresponding switching graph M, a LAP graph must be used as a template. Suppose that G has n cluster vertices and p via candidate vertices. Then G is represented in a $n \times p$ matrix $Template$ which is defined as below:

$$Template[i][j] = \begin{cases} 1 & \text{if} < c_i, v_j > \in E; \\ -1 & \text{if} < v_j, c_i > \in E; \\ 0 & \text{otherwise.} \end{cases}$$

For example, the LAP graph shown in Figure 2 is represented as the matrix shown in Figure 4.

$$Template = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 1 & -1 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Figure 4. Template

Given a chromosome $s = a_1 a_2 \cdots a_n$, its corresponding switching graph M can be obtained by Algorithm 2 below:

Algorithm 2: Decoding

```
for i = 1 to n do
  for j = 1 to p do
    M[i][j] = Template[i][j];
for i = 1 to n do
  if b_i = 1 then
    for j = 1 to p do
      M[i][j] = -M[i][j];
```

For example, the switching graph of chromosome $s = 0011000$ is decoded as the above matrix shown in Figure 5.

$$Template = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -1 & 1 \\ 0 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Figure 5. The switching graph of s=00110000

It can be easily calculated from the above matrix that $via(s) = 2$ because there are only via vertices v_5 and v_7 (their both in-degree and out-degree are not zero). Thus, $f(s) = 5$.

6. Genetic operators

This section details the genetic operators used in KBGA, and focuses on discussing knowledge-based operators.

6.1. Knowledge-based crossover

It is known that there is a problem for the conventional one-point crossover, that is, two reproduced offspring could be less fitter than their parents. Let's look at the conventional one-point crossover for two selected parents shown in Figure 6. Figure 7 shows the two offspring produced by the one-point crossover. It can be calculated that the values of fitness for p_1 and p_2 are both 4, while the values of fitness for their two offspring o_1 and o_2 are 4 and 2 respectively. Hence, the total fitness of the two parents is 8 while the total fitness of the two offspring is reduced to 6.

In order to overcome this problem, a knowledge-based crossover is used in the KBGA. Different from a conventional crossover operator, this crossover is a knowledge-based one which can make sure that the two offspring to be reproduced are fitter than their parents. In this way, the average fitness of the chromosomes in a population has much

more probability to be better than that of the previous generation. As a result, it will contribute to the quick convergence of the genetic algorithm.

The basic idea behind the knowledge-based crossover is: when reproducing two offspring from two selected parents, we identify some good genes and pass these genes to their offspring. By good genes, we refer to the genes which are associated with non-via vertices in the corresponding switching graph. However, there may be some conflicts between the genes from the two parents. An example of such conflicts is that for a particular gene is good one in both of their parents, while in one parent the allele of the gene is '1', while in another parent the allele of the gene is '0'. As a result, we cannot pass the gene the two offspring at the same time. Because of this reason, this crossover passes the good genes from one parent to just one of the two offspring and the rest genes of the offspring are copied from the corresponding positions at the other parent. Similarly, we produce the other offspring. Figure 8 shows an example of the crossover operator. In the figure, $P_1 = 0111000$ and $P_2 = 1010110$ are two selected parents, $O_1 = 0011010$ and $O_2 = 0110110$ are the two offspring produced by the knowledge-based crossover. It can be calculated that the values of the fitness function for two produced offspring, O_1 and O_2, are both 6. As a result, total fitness of the two produced offspring is 10.

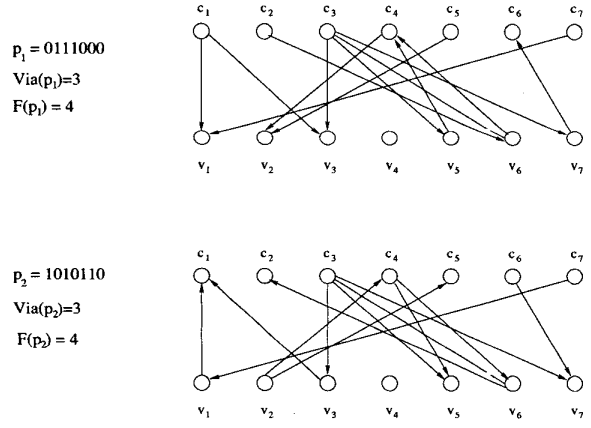


Figure 6. Two selected parents

6.2. Mutation

Mutation is to randomly change an allele of an individual from one alphabet value to another in order to keep diversity in the generations. Since a binary alphabet is used over the

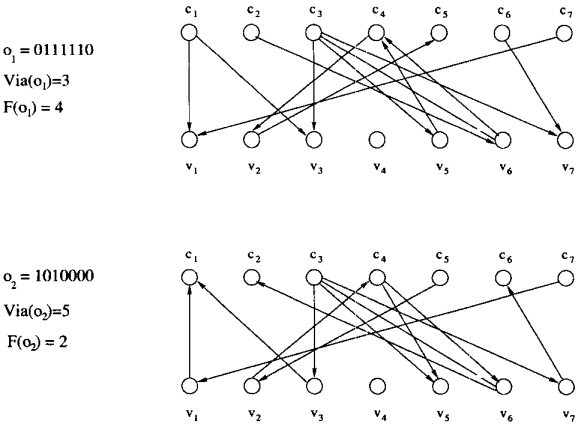


Figure 7. Conventional crossover

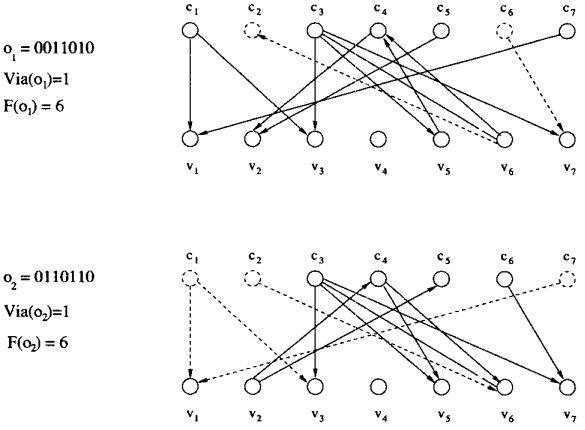


Figure 8. Knowledge-based crossover

constrained via minimization problem, the original allele is exchanged for its complement.

The mutation operator offers the opportunity for new genetic material to be introduced into a population. From the theoretical perspective, it ensures that given any population, the entire search space is reachable. The new genetic material does not originate from the parents and is not introduced into their children by crossover.

6.3. Hill-climbing operator

The hill-climbing operator is used to improve the fitness of an individual. As its name implies, the operator is based on hill climbing technique [3].

This hill-climbing operator is based on the hill-climbing algorithm we presented in [13], which finds a local optimum switching graph from a LAP graph.

7. Implementation and experimentation

The KBGA has been implemented in C on a Pentium 200 personal computer. The program contains about 3000 lines source code.

In order to test the KBGA, we developed a program to randomly generate switching graph problems. For each randomly generated problem we use four different methods: optimal algorithm, hill-climbing algorithm [13], traditional genetic algorithm [14], as well as the KBGA. The optimal algorithm enumerates all the possible layer assignments and outputs the layer assignment requiring minimal number of vias. Due to the inherent high computational complexity of the optimal algorithm ($O(2^n)$, where n is the number of cluster vertices in a switching graph), it is impossible for the optimal algorithm to find the optimal solution for a large-size problem in a tolerable time. Thus, we are not able to know what the optimal solutions for those large-size problems are and how many vias introduced in the optimal solutions. The hill-climbing algorithm is the one we proposed in our preliminary research on the layer assignment problem. The traditional genetic algorithm is the first genetic algorithm applied on the layer assignment problem which does not use knowledge-based crossover or hill-climbing operator.

Firstly, we randomly generate 36 switching graph problems. Then, the optimal algorithm is applied on 27 of those switching graph problems to obtain the optimal solutions (the sizes of the rest 9 switching graph problems are too large to use the optimal algorithm). After that, the hill-climbing algorithm, traditional genetic algorithm and the KBGA are used. The experimental results are shown in Table 1.

In the table, n and p are the number of cluster vertices and the number of via candidate vertices of a randomly generated switching graph, respectively; The $vias$ in the column $Characteristics$ is the number of vias introduced in the switching graph; The $vias$ and $time$ in the columns $Optimal$, $Hill-Climbing$, GA and $KBGA$ are the number of vias and the time spent on the layer assignment for the optimal algorithm, the hill-climbing algorithm, the traditional genetic algorithm and this knowledge-based genetic algorithm, respectively.

It can be seen from Table 1 that the computation time of the optimal algorithm increases dramatically with the increase of the size of switching graph problems. Hence it cannot be used for large-size layer assignment problems. In contrast, the computation speed of the hill-climbing algorithm is very fast. But the quality of the solutions obtained by the hill-climbing algorithm is not satisfied. For all the 27 switching graph problem whose optimal solutions have been identified by the optimal algorithm, the hill-climbing algorithm failed to find the optimal solutions for 15 of them. The traditional genetic algorithm found the optimal solutions for 24 of the 27 switching graph problems and its computation time is in the order of dozens of seconds. The KBGA successfully found the optimal solutions for all the 27 switching graph problems just in seconds.

8. Conclusions

Basing on the switching graph model we presented a knowledge-based genetic algorithm for the layer assignment problem in this paper, with the aim of utilizing domain-specific knowledge to speedup the process of evolution and improving the quality of solutions.

By using the knowledge-based crossover operator we can make sure the process of the evolution of this knowledge-based genetic algorithm is convergent. By using the hill-climbing operator to produce a local optimal, we can not only speedup the computation speed, but also generate some good genes which contribute to find the optimal solution.

Although the computation speed is not as fast as that of the heuristic algorithm, the quality of the solutions obtained by the KBGA is much better than that of the heuristic algorithm. In fact, it is able to find the optimal solutions in most cases.

The unique power of genetic algorithms shows up with parallel computing. Parallel genetic algorithm with information exchange between searches are often more efficient than independent searches. Our future research on the layer assignment will extend to parallel genetic algorithm.

References

[1] K. Ahn and S. Sahni, Constrained via minimization, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(2):273-282, 1993.

[2] F. Barahona, On via minimization, *IEEE Trans. on Circuits and Systems*, 37(4):527-530, 1990.

[3] A. Bundy (ed), *Artificial Intelligence Techniques: A Comprehensive Catalogue*, Springer, Berlin, 1997.

[4] C.C. Chang and J. Cong, An efficient approach to multi-layer layer assignment with application to via minimization, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 18(5):608-620, 1999.

[5] Y.-C. Chou and Y.-L. Lin, A graph-partitioning-based approach for multi-layer constrained via minimization, *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 426-429, 1998.

[6] R. Diestel, *Graph Theory*, Springer, New York, 1997.

[7] J.H. Holand, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Michigan, 1975.

Table 1. Experimental results for the randomly generated switching graph problems

Problem	*Characteristics*			*Optimal*		*Hill-Climbing*		*GA*		*KBGA*	
Index	*n*	*p*	*vias*	*vias*	*time(sec)*	*vias*	*time(sec)*	*vias*	*time(sec)*	*vias*	*time(sec)*
1	12	9	7	2	9	2	0.0002	2	11	2	1
2	12	16	13	7	9	8	0.0004	7	19	7	1
3	14	21	13	7	34	7	0.0005	7	22	7	2
4	58	10	9	-	-	3	0.0012	3	34	3	1
5	34	14	12	-	-	2	0.0013	2	39	2	5
6	28	30	17	10	150842	10	0.0019	10	51	10	5
7	13	49	37	24	17	25	0.0011	25	63	24	8
8	11	28	23	15	4	17	0.0017	15	56	15	4
9	16	22	15	7	136	8	0.0008	7	30	7	2
10	12	11	7	1	9	3	0.0002	1	19	1	1
11	18	21	13	9	493	9	0.0006	9	28	9	2
12	26	25	18	8	11098	11	0.0014	8	67	8	4
13	48	19	16	-	-	6	0.0024	4	52	4	3
14	30	12	14	3	244574	3	0.0010	3	29	3	2
15	38	16	9	-	-	3	0.0012	1	31	1	3
16	38	20	10	-	-	6	0.0013	5	39	5	3
17	43	20	13	-	-	5	0.0022	4	45	4	4
18	16	42	29	20	140	22	0.0011	20	83	20	7
19	20	42	31	18	1505	22	0.0022	19	98	18	8
20	17	27	17	9	274	9	0.0009	9	57	9	4
21	16	9	5	1	133	3	0.0030	1	9	1	1
22	13	19	26	17	18	17	0.0010	17	82	17	6
23	25	25	20	8	6081	10	0.0017	8	60	8	4
24	22	37	24	16	5144	16	0.0019	16	87	16	7
25	10	19	9	7	2	9	0.0002	7	34	7	2
26	19	23	14	7	1096	10	0.0008	8	49	7	3
27	19	28	18	11	767	12	0.0009	11	61	11	4
28	23	28	19	11	8013	12	0.0014	11	70	11	5
29	28	28	19	8	92699	8	0.0019	8	65	8	5
30	8	12	9	4	1	5	0.0002	4	41	4	3
31	42	16	13	-	-	5	0.0015	2	37	2	2
32	30	13	5	2	285311	2	0.0006	2	27	2	1
33	51	15	9	-	-	2	0.0018	1	33	1	2
34	17	10	10	2	48	2	0.0004	2	16	2	2
35	55	18	16	-	-	3	0.0037	3	75	3	4
36	14	58	34	28	6	28	0.0013	28	102	28	9

[8] D.E. Goldbery, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, 1989.

[9] P. Molitor, A hierarchy preserving hierarchical bottom-up 2-layer wiring algorithm with respect to via minimization, *Integration, The VLSI Journal*, 15(1):73-96, 1993.

[10] N.J. Naclerio, S. Masuda, and K. Nakajima, The via minimization is NP-complete, *IEEE Trans. on Computers*, 38(11):1604-1608, 1989.

[11] C.-J.R. Shi, Solving constrained via minimization by compact linear programming, *Proc. IEEE Asia and South Pacific Design Automation Conference*, 635-640, 1997.

[12] C.-J Shi, J.A. Brzozowski, A charaterization of signed hypergraphs and its applications to VLSI via minimization and logic synthesis, *Discrete Applied Mathematics*, 90(1):223-243, 1999.

[13] M. Tang, K. Eshraghian, and H.N. Cheung, An efficient approach to constrained via minimization for two-layer VLSI routing, *Proc. IEEE Asia and South Pacific Design Automation Conference*, Hong Kong, 149-152, 1999.

[14] M. Tang, K. Eshraghian, and H.N. Cheung, A genetic algorithm for constrained via minimization, *Proc. IEEE International Conference on Neural Information Processing*, Perth, 435-440, 1999.