A Genetic Algorithm for Channel Routing using Inter-Cluster Mutation

B. B. Prahlada Rao, L. M. Patnaik and R. C. Hansdah
Department of Computer Science and Automation
Indian Institute of Science
Bangalore - 560 012
India

Abstract

In this paper, we propose an algorithm for the channel routing problem based on genetic approach that uses a new type of mutation, called inter-cluster mutation . The performance of genetic algorithm-based channel router is improved by incorporating problem-specific knowledge into the inter-cluster mutation operators. A solution of the channel routing problem is represented by horizontal non-constraint graph(HNCG). The clusters information in the graph of the routing solution is used in the proposed inter-cluster mutation operators. The inter-cluster mutation integrates the concept of gradient descendent method, implicitly, into the genetic algorithm based channel router. We have implemented the inter-cluster mutation operators in C, and the results of the experiment show improved performance for genetic algorithm-based channel router using inter-cluster mutation.

1 Introduction

Interconnections in VLSI circuits occupy a significant portion (approximately 50-60%) of total chip area. An efficient routing of these interconnections reduces chip size, and improves the reliability of the chip design. Hence, routing has received a great deal of attention from the researchers. The concept of channel routing (CHR) was first proposed in [1]. Since then various algorithms have been proposed for channel routing [2,3,5]. A *channel* is a rectangular region with terminals at the horizontal sides of the region. A *net* in a channel is the set of terminals to be interconnected. The objective of a channel router is to interconnect the terminals in each net of the channel, using minimum number of tracks, so as to reduce the total chip area. Other parameters such as the number of vias and the number of doglegs are considered as secondary measures while evaluating the quality of a routing solution.

It is well-known that the channel routing problem is NP-complete[4]. Hence, several algorithms have been proposed to find approximate solution to the channel routing problem. Most of the channel routing algorithms [2,5] are sequential in nature, i.e., they work either on one track of the channel or on one possible routing solution at a time. Probabilistic algorithms such as simulated annealing are also used to solve the channel routing problem. Simulated annealing based channel routers [3] start with an initial solution and try to improve the solution by perturbing it to its neighbourhood at a time, following the principles of physical annealing in metallurgy. Even though methods have been proposed to parallelize simulated annealing, Simulated annealing is inherently sequential in nature. The genetic algorithm-based channel router [9] is inherently parallel in nature. This methodology starts with a set of initial solutions. At every time, it combines the features of two solutions, called parents, to produce two new solutions, called offspring, which inherit the features of the parents, and thereby this method tries to search for new potential solution(s).

Application of genetic algorithm to a problem requires an encoding scheme, an evaluation function and a set of genetic operators, viz., selection, crossover, and mutation operators. The normal binary mutation used in channel routing based on genetic approach [9] has a high tendency of randomly merging and/or breaking the clusters. Hence, the algorithm takes longer time to converge to optimal routing solution(s). In this paper, a set of intercluster mutation(ICM) operators is proposed that use the clusters information of the

0-7803-1899-4/94 $4.00 ©1994 IEEE

Algorithm Traditional_GA
Generate an initial population of size POPSIZE;
for generation = 1 to MAXGEN **do**
  Calculate fitness statistics for each individual in the population;
  Select POPSIZE parents probabilistically based on the individual's fitness;
  for generation i = 1 to POPSIZE/2 **do**
    Pair two parents randomly without replacement;
    Crossover the parents based on $P_{crossover}$ and produce two new offsprings;
    Mutate each offspring based on Mutation rate;
  endfor
endfor

Figure 1: Traditional Genetic Algorithm

HNCG graph of the chromosome. ICM operators do not randomly disrupt the optimal partition information gained by the algorithm over the previous generations. Hence, ICMs give better performance results for the genetic algorithm-based channel router compared to normal binary mutation.

The rest of the paper is organized as follows. Section 2 gives an overview of genetic algorithms. Section 3 explains the channel routing problem. In section 4, we describe how the channel routing problem is solved using genetic approach. This section also explains genetic operators including inter-cluster mutation operators, and the evaluation function used . Results and conclusions are presented in section 5.

2 Overview of Genetic Algorithms

Genetic algorithms(GA) are stochastic search algorithms based on biological evolution models, whose main advantage lies in its robustness of search and problem independence. The basic concepts of GA were developed by Holland in 1975[6]. The algorithm operates on a population of individuals which represent points in the search space. Each individual has some fitness value or figure of merit and is measured by an evaluation function. The approach of the algorithm is to explore the search space and to discover better solutions by allowing the individuals to evolve over time. The time steps for evolution in a GA are called generations. The traditional genetic algorithm is shown in fig. 1.

The algorithm's main loop is executed once for each generation. For each generation, the algorithm calculates the fitness value for each individual in the population, selects fittest individuals for production, and reproduces offsprings using crossover and mutation operations. **Selection**, **crossover** and **mutation** are the basic operators of GA. In order to solve a problem using GA, the following five components are required:

1. A genetic encoding scheme to represent the solution space of the problem.
2. A mechanism to create an initial population of solutions.
3. An evaluation function that plays the role of the environment that rates the solutions: Evaluation functions are problem-dependent. They reflect the constraints on the problem in the sense that, for each solution, the function evaluates the solutions to show how good or bad the solutions are. While the good solutions are evaluated "high", the bad ones are evaluated "poor". Therefore, the design of an evaluation function is vital for proper functioning of genetic algorithms.
4. Selection of the parameter values used in GA (POPSIZE, Probabilities of genetic operators, MAXGEN ..etc).
5. A set of genetic operators, viz., **selection**, **crossover**, **mutation** and a few others as required by the application: These operators have to be defined to suit the application and the power of the GA lies in the proper selection of these operators.

3 Channel Routing Problem

In this section, we present a mathematical formulation of channel routing(CHR) problem. A horizontal channel is assumed. The routing is done in two layers, one for horizontal segments and the other for vertical segments. A net is a set of terminals to be interconnected and is denoted by n_i. Let **l** be the length of the channel and N be the set of nets to be routed. The set of terminals on the top(bottom) of the channel are denoted by $T(B)$.

Let
$T = \{t_1, t_2, ..., t_l\}$,
$B = \{b_1, b_2, ..., b_l\}$,
$N = \{n_1, n_2, ..., n_k\}$,
where
$n_i \subset T \bigcup B$;
t_i or b_i can take values 0, L, R or any value from the set N;
0 represents that the terminal is not connected to any net;
L represents that the terminal is connected to a net entering from the left side of the channel;
R represents that the terminal is connected to a net entering from the right side of the channel.

We say that there exists a **horizontal constraint** between nets n_i and n_j if the horizontal segments of these nets overlap. Therefore, these nets cannot be placed in the same track in the channel. Horizontal constraints can be represented using the horizontal non-constraint graph(HNCG), HNCG = (N,E), where N is the set of nets, and an undirected edge (n_i,n_j)$\in$E indicates that the horizontal spans of nets n_i , n_j do not overlap.

A **vertical constraint** states that, at any column along the channel length, the horizontal segment of a net connected to the top terminal should be placed above the horizontal segment of the net connected to the bottom terminal.

Splitting the horizontal segment of a net into more than one horizontal segment is called **doglegging**.

3.1 Objective of channel routing

The objective of CHR is to assign tracks in the channel to the given set of nets using a minimum number of tracks satisfying the following Constraints:

1. In every track, no horizontal constraints are violated .
2. No violation of vertical constraints at any column along the channel.
3. Interconnections are restricted to two layers.

4 Problem Formulation

The channel routing problem is (i) to find a partition of the HNCG graph with minimum number of disjoint clusters(a subgraph of HNCG, which is complete), and (ii) to assign each cluster to a different track without a vertical constraint violation. It is to be noted that any one-to-one mapping of clusters to tracks would not result in a horizontal constraint violation, since clusters are disjoint, and each cluster is complete. A partition of a graph G is a set of non-overlapping clusters of G.

In this paper, a genetic approach has been used to find a solution for the channel routing problem. In this approach, we take a collection of randomly generated graphs. Each graph has the same number of vertices as that of the HNCG graph of the channel routing problem. Each connected subgraph of G is treated as a complete subgraph, and the resulting clusters of G are disjoint. The set of clusters of G so obtained is considered as a possible partition for the HNCG graph. Using the HNCG graph, the number of horizontal constraint violations are found out among the nets in each clusters of G. Then each cluster of G is assigned a label representing the track number in which nets of the cluster are routed. After assigning the tracks to each cluster of G, the total number of vertical constraint violations are found out. The fitness value of G is determined using an evaluation function, whose arguments are number of horizontal constraint violations, number of vertical constraint violations, and the number of clusters. The fitness value of each graph G in the

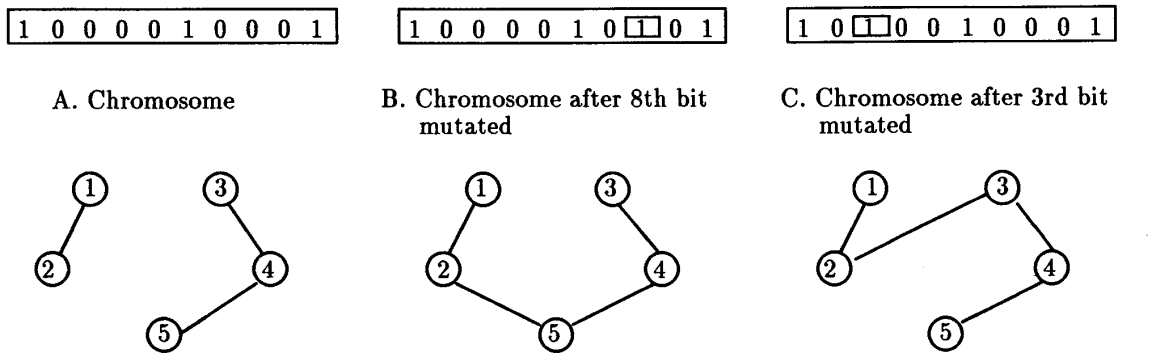

Figure 2: Effect of Random Alteration of Bits in the Chromosome

collection is evaluated likewise. Then the genetic operators selection, crossover and mutation are applied to the collection of graphs resulting in a new collection of graphs. The process of evaluation, selection, crossover and mutation are repeated till the terminating condition is reached. The GA operators are explained below.

4.1 Encoding scheme for CHR problem

The encoding scheme is the same as that discussed in[9]. Since the adjacency matrix representation of a graph is always symmetric, the lower triangle of the adjacency matrix is represented as a linear bit string; we thus get a unique encoding for the graph. We encode a k-node graph using k(k-1)/2 binary bits. The chromosome is constructed from the adjacency matrix as follows.

for i = 2 to number-of-nets
  for j = 1 to (i - 1)
    $chrom[\frac{(i-1)(i-2)}{2} + j] = 1$ if (i, j)th element of the adjacency matrix is equal to 1
    $= 0$ otherwise

4.2 Crossover

Crossover is done on two selected strings, called parents. The resulting strings of the crossover are called offsprings. We have used single point crossover in our GA implementation for CHR. In single point crossover [7], we select a point within the chromosome (called crossover point), which divides both the parents into two segments each. Offsprings are generated by mutually exchanging the corresponding segments of the parent chromosomes after the crossover point.

4.3 Mutation

The **mutation** operator is a method by which a solution is perturbed. The mutation operator is meant for reducing the allele loss in the chromosome. This operator randomly alters some of the bits in the chromosome, and is acceptable for binary chromosomes in function optimization problems, where the chromosome represents the parameters of the function. But when the chromosome represents a graph [9], as in the case of channel routing problem, normal mutation scheme causes random breaking and/or merging of clusters. An example to illustrate this phenomena is shown in fig.2. This normal mutation causes loss of the optimal partition information gained by GA, over the previous generations. Hence, the algorithm takes more number of generations for finding the optimal routing configuration. We propose a set of mutation operators called inter-cluster mutation (ICM) that overcomes the defects of normal mutation in the genetic algorithm based channel router.

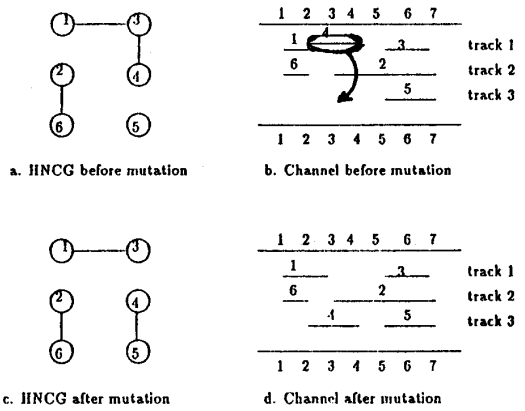

Figure 3: Illustration of MoveNet-Inter-Cluster Mutation Scheme

4.3.1 Inter Cluster Mutations (ICMs)

The **ICMs** make use of the cluster information of the chromosome, i.e. , the nets in the cluster, number of horizontal constraint violations among the nets within a cluster, the track number assigned to the cluster, etc. . Many different ICMs are possible depending upon the choice of source cluster, destination cluster, and the mechanism of moving nets between the clusters. Some of the possible ICMs that we have implemented are as follows:

1. **MoveNet:** Select a cluster having maximum number of horizontal constraint violations (HCVs) as source cluster, and a cluster having zero or minimum number of HCVs as destination cluster. Select a net within the source cluster that has maximum number of HCVs with the remaining nets in this cluster. Move the selected net form the source cluster to the destination cluster. This is equivalent to moving a net from one track to another track in the channel. This is illustrated in fig.3. This operator helps to reduce the number of horizontal constraint violations in a routing solution.

2. **ExchangeNets**: Select randomly two clusters in the graph representing the chromosome. Select a net from each cluster and mutually exchange these selected nets between the clusters. This operator also helps to reduce the number of horizontal constraint violations in a routing solution.

3. **MergeCluster**: Select a cluster in the graph representing the chromosome having only one net. Move the net of the selected cluster into another cluster. This operator helps to reduce the number of tracks in a routing solution.

4. **BreakCluster**: Select a cluster having maximum number of HCVs in the graph representing the chromosome. In the cluster, select the net k that has the highest number of HCVs, and select the set of all the nets(S) that have no HCVs with k. Remove net k and the nets in the set S from the cluster and form a new cluster having net k and the nets in set S. This mutation operator reduces the number of HCVs by increasing the number of tracks in a routing solution.

5. **VerticalSwap**: Select a column C along the channel length having the vertical segments overlapped at C. Swap the tracks of the nets that are connected to the top and bottom terminals at C. This mutation helps to reduce the number of vertical constraint violations in a routing solution.

These ICM schemes try to find, from a given graph, a nearest graph that has higher fitness value. This, in turn, will mutate a chromosome of lower fitness value to a chromosome of higher fitness value. Thus, the ICMs integrate the gradient descent method implicitly into the GA, i.e., increase the fitness value of each chromosome by performing a selected mutation heuristic as discussed above.

In order to incorporate the randomness of mutation, One of the ICM schemes can be selected at random and applied to the given graph. This can be further improved by maintaining probabilities of executing

```
Decode( Chromosome) ;
If(W == 1) or (W < W_l)
    Use BreakCluster mutation of ICM.
else if (TC == 0) & (W > W_e)
    Use MergeCluster mutation of ICM.
else if( H > 0 )
    Use MoveNet or ExchangeNets mutations of ICM.
else if( H == 0 ) & ( V > 0 )
    Use VerticalSwap mutation of ICM.
Chromosome= MakeChrom(graph).
```

Figure 4: Mutation_ICM

each mutation scheme based on its merit. For example MoveNet and ExchangeNets mutations have to be performed more freequently compared to MergeCluster, BreakCluster and VerticalSwap mutations. We give a simple implementation by which the above various ICMs are integrated, using the function *Mutation_ICM* which applies one of the ICM operator depending on the charecteristics of the chromosome to be mutated. The Mutation_ICM is shown in fig.4 and the functions *Decode*, *MakeChrom* and some of the parameters used are explained below.

1. **Decode (chormosome):** This gives the HNCG graph of the chromosome, the number of clusters W of the graph, the total number of horizontal constraint violations H, and the total number of vertical constraint violations V in the HNCG graph that corresponds to the chromosome.
2. **Total Constraints(TC)** = H + V.
3. **Limits of W:** Depending on the net data to be routed, it is possible to find the lower bound for the value of W, W_l and expected value of W, W_e .
4. **MakeChrom(graph):** This transforms the graph information to a chromosome.

Due to random generation of the chromosome it is likely that all nets are interconnected and formed into one cluster only. In such cases, BreakCluster operator splits this large cluster and reduces the number of constraint violations. Whenever the number of tracks in the routing solution are higher than the expected value, MergeCluster helps to reduces the number of tracks in the routing solution. If the routing solution is invalid, MoveNet and ExchangeNets try to transform the solution into a valid solution. If the routing solution has only vertical constraints, it is apt to use VerticalSwap operator.

4.4 Evaluation function

The evaluation function that is used to evaluate the quality of a routing solution in the GA is shown below.

$$f(w, H, V) = \lambda_1 - \lambda_2 w^2 - \lambda_3 H - \lambda_4 V$$

where w = Number of tracks (width) of the routing solution.
H = Total number of horizontal constraint violations in the routing solution
V = Total number of vertical constraint violations in the routing solution,
and $\lambda_1, \lambda_2, \lambda 3$ and λ_4 are constants.

This evaluation function is a slight variant of the evaluation function proposed in [9]. Proper selection of the constants affects the speed of convergence and quality of the final solution.

5 Implementation and Analysis of Results

The proposed algorithm has been implemented in C. The algorithm is tested on many benchmark example problems and it is observed that the optimal configuration is reached in most of the cases. We demonstrate

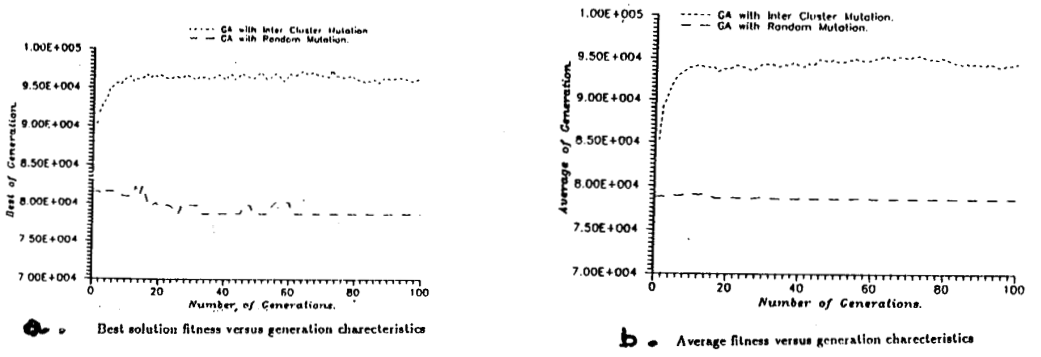

Figure 5:

the behavior of the algorithm on a routing problem, with 24 nets and having a channel length of 25. The optimal routing width of the above problem is 6. The fitness value of the optimal solution is 99640.0 with λ_1=100000, λ_2=10, λ_3=200 and λ_4=200. After many runs of the CHR algorithm, the constants λ_2, λ_3 and λ_4 in the evaluation function are empirically fixed as 10, 200 and 200 respectively. The superiority of the ICMs over normal binary mutation are experimentally demonstrated in fig.5a and fig5b.The graph in fig.5 shows the fitness value of the best solution found till the generation. The graph in fig.5 shows average fitness versus generation charecteristics. Fig.5 shows how the average fitness value of the entire population progresses with every iteration (or generation) of the GA for CHR problem. The increase of the average fitness value of the population as the algorithm progresses is an indication that all the routing solutions in the population are converging to the optimal routing solutions. In this investigation, we have used multi terminal nets. The results on example channel routing problems are not shown here due to space restrictions.

References

[1] A.Hashimoto and J.Stevens. Wire Routing by Optimizing Channel Assignment Within Large Apertures, Proceedings of 8th ACM/IEEE Design Automation Conference, 1971, pp. 214-224.

[2] M.Burstein and R.Pelavin. Hierarchical Wire Routing, IEEE Tr. on CAD, Vol CAD-2, Oct 1983, pp. 223-234.

[3] H.W.Leong , D.F.Wong and C.L.Liu. A Simulated Annealing Channel Router, Proceedings of IEEE Intl. Conf. on CAD (ICCAD) 1985, pp. 226-228.

[4] T.G.Szymanski. Dogleg Channel Routing is NP-Complete, IEEE Tr. on CAD, Vol. CAD-4, No-1, January 1985, pp. 31-41.

[5] Uzi Yoeli. A Robust Channel Router, IEEE Tr. on CAD, Vol.10, No-11, Feb 1991, pp. 212-219,

[6] John Holland. Adaptation in Natural and Artificial Systems, Ph.D. Thesis, MIT 1975.

[7] David E.Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Publishing Co. Inc., 1989.

[8] J.P.Cohoon et al., . Distributed Genetic Algorithms for the Floorplan Design Problem, IEEE Tr. on CAD, Vol.10, April 1991, pp. 484-492.

[9] B.B.Prahlada Rao, L.M.Patnaik and R.C.Hansdah. Parallel Genetic Algorithm for Channel Routing Problem, Proc. of the IEEE third Great Lake Symposium on VLSI Design, Kalamazoo, Michigan March 5-6, 1993, pp. 69-70.