Extended Distributed Genetic Algorithm for Channel Routing

B. B. Prahlada Rao*and R. C. Hansdah †
Computer Science and Automation Department
Indian Institute of Science
Bangalore - 560 012
India

Abstract

In this paper, we propose a new parallel genetic algorithm (GA), called Extended Distributed Genetic Algorithm (EDGA), for channel routing problem. The EDGA combines the advantages of previous parallel GA models , viz., master/slave GA model and distributed GA model. In EDGA, the root processor executes the conventional genetic algorithm with global selection on total population and the remaining processors execute conventional genetic algorithm with local selection on subpopulations. After certain number of generations, the total population on the root processor and the subpopulations on the remaining processors are interchanged, and the process is repeated till terminating conditions are reached. This incorporates features of both global and local selection in the proposed EDGA. The EDGA is designed to obtain good speedup, global optimal solution, and full utilization of the parallel system. We have implemented master/slave GA, distributed GA, and the proposed EDGA in C on a transputer-based parallel MIMD machine and compared their performance. It is found that the EDGA achieves higher speedup than both master/slave GA, and distributed GA.

1 Introduction

In the physical design of VLSI circuits , the problem of circuit layout is quite difficult, and there exists no known polynomial time algorithm to obtain optimal solution. The layout problem is broadly divided into placement and routing. In placement, all the components are placed on the chip, and in routing, all the components are interconnected. The routing problem is further divided into global routing and detailed routing. Global routing roughly determines the regions through which the wire passes and detailed routing determines the path of every wire precisely in the region. Channel router is a type of detailed router. 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. A *net* in a channel is a set of terminals to be interconnected. The objective of a channel router is to interconnect all the terminals in each net of the channel using minimum number of tracks, and the problem is shown to be NP-complete [4]. Hence, several algorithms have been proposed to find approximate solution to the channel routing problem.

Most of the channel routing algorithms [2,3,5] 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 [8], it is inherently sequential in nature. The genetic algorithm-based channel router [12] is inherently parallel in nature. This methodology starts with a set of initial solutions. At each iteration, it combines the features of a pair of solutions, called parents, to produce two new solutions, called offsprings, which inherit the features of the parents, and thereby this method tries to search new potential solutions using the principles of biological evolution.

The unique power of genetic algorithms shows up with parallel computers. Parallel searches(i.e., parallel

*email: plad@csa.iisc.ernet.in
†email: hansdah@csa.iisc.ernet.in

GAs) with information exchange between the searches are often more efficient than independent searches. Thus parallel genetic algorithms (PGA) combine the speed of parallel machine and the advantage of inherent parallelism available in the GA. Easiest way to utilize the inherent parallelism of a sequential GA is by evaluating the fitness of the offsprings in parallel [9]. In this approach (master/slave GA) the root processor(master) selects the parents for the next generation and generates the offspring. The slave or worker processors perform mutation on each offspring and evaluate the objective function(fitness) value of the offsprings. Although this is straightforward, it is not always advantageous with respect to the communication time. Parallel MIMD computers suitable for coarse-grain parallelism are characterized by a low communication bandwidth. Most of the existing parallel variants of GA are based on the concept of distributed subpopulations (distributed GA).

A new parallel genetic algorithm for channel routing problem, called Extended Distributed Genetic Algorithm(EDGA), is proposed in this paper. In EDGA, we use the concept of distributed subpopulations [10] to achieve good speedups, and the concept of global selection strategy [9] to obtain better optimal solution(s). The root processor executes the GA with the global selection on the entire population, whereas the processing elements(PE) operate GA on the corresponding subpopulations with local selection. At a predetermined interval, the evolved populations of root and the PEs are exchanged. This cycle is repeated for specified number of cycles(called epochs) or till terminating condition is reached. The combination of locally adapted subpopulations with the globally adapted total population offers the advantages of a fast search within the optimal search regions and better optimal solutions. The EDGA proposed is simple, efficient, and can easily be implemented on the available parallel hardware. The proposed algorithm obtains good speedup and it balances the load on all the processors so as to ensure good performance of the parallel system. We have implemented the master/slave GA, the distributed GA, and the EDGA for channel routing problem on a transputer-based MIMD machine, called PARAM, and compared their performance.

The rest of the paper is organized as follows. Section 2 gives an overview of genetic algorithms. Section 3 explains the channel routing problem. Section 4 presents how the channel routing problem is solved using genetic algorithm followed by a discussion of the genetic operators and the evaluation function used for the channel routing problem. Section 5 presents the extended distributed genetic algorithm. Results and conclusions are presented in Section 6.

Algorithm Sequential_GA
Generate an initial population of size PopSz;
for generation = 1 to MAXGEN **do**
  Calculate fitness statistics for each individual in the population;
  Select PopSz parents probabilistically based on the individual's fitness;
  for generation i = 1 to PopSz/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: Sequential Genetic Algorithm

2 Overview of Genetic Algorithms

Genetic algorithms(GA) are stochastic search methods based on biological evolution models, whose main advantage lies in the 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 conventional Sequential Genetic Algorithm(SGA) 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 reproduction, 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:* Various representation schemes are possible, viz., binary representation, integer representation, matrix representation etc.
2. *A mechanism to create an initial population of so-*

lutions: Initial population of solutions is generated randomly.

3. *An evaluation function that plays the role of the environment and rates the solutions:* Evaluation functions are problem-dependent. They reflect the constraints of the problem in the sense that, for each solution, the function evaluates the solutions to show how good or bad the solutions are.

4. *Selection of the parameter values used in GA (PopSz, Probabilities of genetic operators, MAX-GEN ..etc.):* There is no rigid rule for selecting these parameter values. The genetic algorithm has to be run many times to find suitable values of these parameters for the problem at hand or else the parameters have to be estimated using meta-GA concept [11].

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. The **selection** operator selects PopSz individuals from a population of PopSz individuals dropping some individuals with lower fitness value and taking more copies of individuals having higher fitness value. This incorporates the Darwin's survival of the fittest concept in the GA. The **crossover** operator combines features of two parent structures to form new offsprings. The **mutation** operator randomly alters few bits in each chromosome.

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. Net i 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 $L_i(R_i)$ be the leftmost(rightmost) column in the channel for the net n_i. Then the interval (L_i, R_i) is known as the *span* of the net n_i.

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.

3.1 Horizontal constraints

We say that there exists a horizontal constraint violation between nets n_i and n_j if the horizontal spans 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 segments of nets n_i, n_j do not overlap.

3.2 Vertical constraints

We say that there exists a vertical constraint violation at a column in the channel if, the vertical segments of nets connected to top and bottom terminals at the column overlap. Vertical constraint violation can be avoided at a column if the horizontal segment of the net connected to the top terminal at the column is routed in a track that is above the track in which the horizontal segment of the net connected to the bottom terminal at the column is routed.

3.3 Objective of channel routing

The objective of channel routing problem 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. At no column along the channel length, vertical constraints violated.
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 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[12]. 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. In normal mutation, a bit in the chromosome is randomly selected and altered. When the chromosome represents a partition of a graph [12], normal mutation scheme causes random breaking/merging of clusters represented by the chromosome. This normal mutation causes loss of the optimal partition information gained by GA, over the previous generations. We have implemented the graph-based inter-cluster mutation operators of [13].

4.3.1 Inter Cluster Mutation Scheme(ICM)

The **ICM** makes use of the cluster's information of the chromosome, i.e. , the nets in the cluster, number of horizontal constraints among the nets within a cluster, the track number assigned to the cluster etc. ICM moves nets between clusters.

Some of the ICM strategies are as follows:

1. **MoveNet**: Select a cluster in a chromosome having maximum number of horizontal constraint violations (HCV's) as source cluster and a cluster having zero or minimum number of HCV's as destination cluster. Select a net within the source cluster that has maximum number of HCV's with the remaining nets in this cluster. Move the selected net form the source cluster to the destination cluster.

2. **ExchangeNets**: Select randomly a source and a destination cluster in a chromosome. Select a net from each of these clusters. Mutually exchange the selected nets between the source and the destination cluster.

3. **MergeCluster**: Select a cluster in a chromosome having only one net. Move the net of the selected cluster into another cluster.

4. **BreakCluster**: Select a cluster having maximum number of HCVs in a 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 scheme 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 connected to the top and bottom terminals at C. This mutation scheme helps to reduce the number of vertical constraint violations.

In order to incorporate the randomness of mutation, one of the ICM schemes can be selected at random and applied to the given graph. The ICM is implemented by applying one of the ICM operator discussed above depending on the charecteristics of the chromosome to be mutated. We have implemented inter-cluster mutation as discussed in [13].

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 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 the same as the evaluation function proposed by Rao et al. [12]. Proper selection of the constants improves the convergence speed and quality of the final solution.

5 Parallel Genetic Algorithms

In the master/slave parallel GA model [9], the master performs the selection and crossover operations. The children generated by the master are sent to the slaves. The slaves perform the mutation operation and the corresponding fitness value evaluation on these children. The resultant children along with their fitness values are sent back to the master. The special feature of this model is that the selection process is done on the total population. Hence, the global information of the population is not lost and the optimal solution found will be of global nature. In this model, the communication overhead is linearly proportional to the product of population size and number of generations run. The computational gain obtained in master/slave parallel model is reduced due to the large communication overhead.

In the DGA model, the population is divided into subpopulations, each of which is placed on one processor element. Each processor element(PE) runs the SGA on its subpopulation. The selection done by each PE is local to its subpopulation only. After a finite number of generations, called migration interval, the best solutions of each PE are migrated to the neighbouring processors. Though the DGA model gives better speedups than the master/slave model, the migration of the best individuals creates some communication overhead and, as a result, the speedups obtained are not linear [12]. Because of the local selection, the probability that the obtained solution may not be globally optimal is high.

The proposed EDGA is an extension of the idea of DGA proposed in [10]. The algorithm is illustrated in fig.2. Some of the terms used in the EDGA are explained below:

1. Total population: The population of size PopSz placed at the root processor.
2. Subpopulation: The total population is divided into subpopulations each of size $SubPopSz = \frac{PopSz}{Z}$, where Z= No of PEs. These subpopulations are placed on the PEs.
3. Merging subpopulations: The evolved subpopulations from all the PEs are sent to the root. The subpopulations received are merged and a new total population is generated.
4. Global selection: Selection of strings from total population.
5. Local selection: Selection of strings from subpopulation.
6. Terminating conditions :Terminating condition can be reaching of known optimal fitness value, or running of the GA for fixed number of generations, or average value of population crossing a specified value .. etc.

5.1 Extended Distributed Genetic Algorithm(EDGA)

The EDGA starts with an initial population P_0, randomly generated on the root. A copy of P_0 is divided into subpopulations (sP_{01}, sP_{02}, sP_{03} .. .), and are distributed to the PEs. Now the root runs the conventional SGA on P_0. PEs run the conventional SGA for

specified number of generations(G) on $sP_{0i}s$ ($1 \leq i \leq$ No. of PEs) and send an interrupt to the root. The evolved populations of P_0 and sP_{0i}s are termed as P_0' and sP_{0i}'s respectively. Now all subpopulations sP_{0i}' are merged to form a new populationP_1 on the root. The evolved population P_0' of the root is divided into subpopulations $sP_{1i}s$ and are distributed to the PEs. Now root and PEs run SGA on P_1 and $sP_{1i}(1 \leq i \leq$No. of PEs) respectively. The algorithm repeats this process for K(pre-determined) cycles or till the terminating condition is reached.

On all PEs, the selection is local, i.e., it selects strings from its subpopulation only, and hence, the EDGA does not lose temporarily unfavorable genetic information too early. Since this suboptimal information is preserved for some time, the good genetic components (if any) in these suboptimal solutions are given a chance to contribute their good genes for the global optimal solution in the future evolutions. At the same time, the root processor performs SGA with the selection strategy on the total population(called global selection), and this helps the algorithm in not getting trapped at a local optimum.

The combination of locally adapted subpopulations with the globally adapted population offers EDGA the combined power of converging to global routing solution with a good speedup on the MIMD machine. The root processor is fully engaged all the time, and hence, this contributes to good utilization of the parallel machine.

In our proposed EDGA, emphasis is given for the global selection criterion, speedup achievable, and maximum utilization of the root processor. The advantages of EDGA are: (i) it balances the load on the root and the PEs, (ii) it has the useful features of the DGA model, and the global selection feature of the master/slave model. The working of EDGA is shown in fig.??.

Algorithm Extended Distributed_GA

Randomly generate an initial population P_0
of size PopSz;
Divide the population into Z subpopulations of
size SubPopSz;
Copy one subpopulation to each PE;
Send the parameter values to all the PEs;
for C cycles
  On root run SGA on P_0 until all PEs interrupt;
  dopar(on PEs)
    for generation = 1 **to** G **do**
      Run Sequential SGA on subpopulation;
    endfor
    Send interrupt to the root processor;
    Send evolved subpopulation to **root**;
  endpar
  Make population P_1 with evolved subpopulations;
  Make subpopulations with evolved P_0;
  Send one subpopulation to each PE;
  $P_0 = P_1$;
endfor (for cycles)

Figure 2: Extended Distributed Genetic Algorithm

6 Implementation and Analysis of Results

For the implementation, and demonstration of the GA-based parallel channel router, we have taken a channel routing problem of 24 nets and having channel length of 25. The parameters PopSz, $P_{crossover}$, $P_{mutation}$ to GA are 32, 0.2, and 0.9 respectively. The total number of generations in master/slave model is 96. In DGA and EDGA models, the migration interval is 24, and the number of epochs is 4. After many runs of the sequential genetic algorithm-based channel router, the constants for the evaluation function λ_1, λ_2, λ_3 and λ_4 are empirically fixed as 100000, 10, 200 and 200 respectively.

The master/slave GA model, distributed GA model and EDGA models were implemented in C on a transputer-based parallel MIMD machine, called PARAM. These parallel GAs were run/executed on 1, 2, 4, 8 and 16 PEs, configured as zero dimensional, one dimensional, two dimensional, three dimensional and four dimensional hypercubes respectively. The computation and communication times for each of the these paralle GAs were measured and are shown in tables 2 to 6.

The communication times were measured both including and excluding waiting time. Communication time excluding waiting time refers to the actual time required for the message transfer between the sender and receiver tasks, and the communication time including waiting time refers to the message transfer time and the time overhead involved in waiting till both the sender task and receiver task gets synchronized.

The computation and communication times (excluding waiting time) for master/slave, DGA, and EDGA are shown in tables 2, 3, 4 respectively. Tables 5 and 6 show computation and communication times(including waiting time) for the DGA and EDGA models respec-

tively. We compared the speedups of all the above models in tables 1 and 7. From tables 3, 4, 5, and 6, it is clear that the computation times in EDGA and DGA are much higher than the communication times. Also the communication time in DGA seems to increase as the number of processors is increased; whereas it decreases in EDGA.

The speedup obtained in the EDGA model always surpassed the speedup obtained in the master/slave model. When the waiting times are included in communication time measurement, it can be observed(table 7) that the speedup for the EDGA model is better than that of the DGA model, when the number of processors exceeded 5. But when the waiting times are excluded in communication time measurement(table 1), the speedup for the EDGA model is always better than that of the DGA model.

Table 1 Speedup Charecteristics

No. of PEs	Master/Slave Model	Dga Model	Edga Model
1	1.000	1.000	1.000
3	0.920	1.972	1.974
5	1.392	3.934	3.959
9	1.965	7.636	7.956
17	2.365	14.478	15.5964

Table 2 Timings in master/slave model

No. of PEs	Comput. Time(Sec.)	Commun. Time(Sec.)	Total Time(Sec.)
1	508.77	0.00	508.77
3	269.39	283.48	552.87
5	132.69	231.79	364.49
9	68.70	190.16	258.86
17	35.61	179.46	215.08

Table 3 Timings in DGA model

No. of PEs	Comput. Time(Sec.)	Commun. Time(Sec.)	Total Time(Sec.)
1	506.23	0.00	506.23
3	252.63	0.0199	252.65
5	128.63	0.0332	128.665
9	66.24	0.0469	66.289
17	34.17	0.1567	34.326

Table 4 Timings in EDGA model

No. of PEs	Comput. Time(Sec.)	Commn. Time(Sec.)	Total Time(Sec.)
1	460.73	0.00	460.73
3	233.05	0.305	233.355
5	116.04	0.328	116.368
9	57.54	0.369	58.016
17	29.07	0.477	29.547

Table 5 Timings in DGA model(with waiting time included in the Communication time measurement)

No. of PEs	Comput. Time(Sec.)	Commn. Time(Sec.)	Total Time(Sec.)
1	506.23	0.00	506.23
3	252.63	0.0359	252.66
5	128.63	31.90	160.53
9	66.24	83.86	150.10
17	34.17	117.12	151.29

Table 6 Timings in EDGA model(with waiting time included in the Communication time measurement)

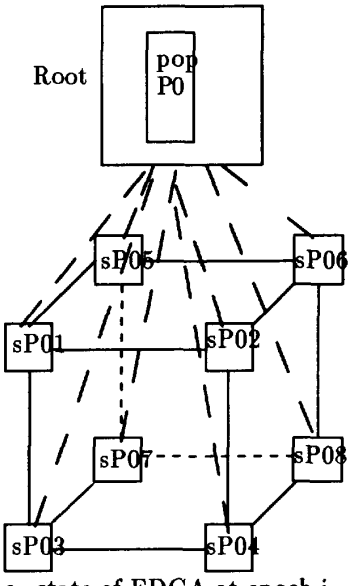
No. of PEs	Comput. Time(Sec.)	Commn. Time(Sec.)	Total Time(Sec.)
1	460.73	0.00	460.73
3	233.05	125.27	349
5	116.04	99.58	215.62
9	57.54	35.72	93.26
17	29.07	36.16	65.23

Table 7 Speedup Charecteristics(with waiting time included in the communication time measurement)

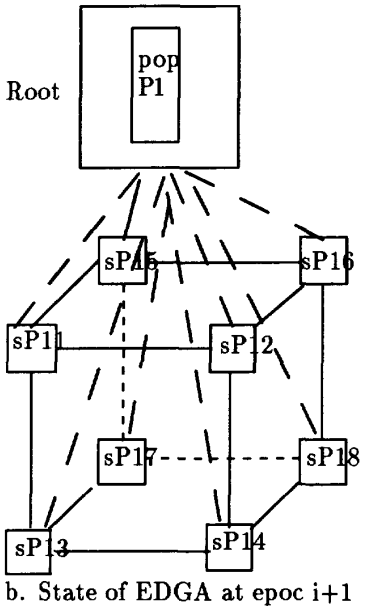
No. of PEs	Master/Slave Model	Dga Model	Edga Model
1	1.000	1.000	1.000
3	0.920	2.000	1.32
5	1.392	3.153	2.14
9	1.965	3.372	4.91
17	2.365	3.346	7.06

7 Acknowledgments

We are very thankful to Prof. L.M.Patnaik, for his valuable guidance and suggestions, throughout this work, without which it would not have been possible to bring out this paper.



a. state of EDGA at epoch i



b. State of EDGA at epoc i+1

Figure 3: Illustration of Extended Distributed GA.

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] S. A. Kravitz and R. A. Rutenbar. Placement by Simulated Annealing on a Multiprocessor, IEEE Tr. on CAD, Vol. CAD-6, No. 4, July 1987.

[9] R. Kottkamp. Nicht-lineare Optimierung Unter Verwendung Verteilter, paralleler Prozess in einem Local Area Network(LAN). Master thesis, University of Dortmund, Dortmund, FRG, February 1989.

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

[11] J. J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms, IEEE Tr. on Systems, Man and Cybernetics, Vol SMC-16, No. 1, Jan/Feb. 1986, pp. 122-128.

[12] 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.

[13] B. B. Prahlada Rao, L. M. Patnaik and R. C. Hansdah. A Genetic Algorithm for Channel Routing Using Inter-Cluster Mutation, Submitted for Publication.