IEEE International Conference on Evolutionary Computation, Indianapolis, April 1997

A Multi-Layer Detailed Routing Approach based on Evolutionary Algorithms

Nicole Göckel, Rolf Drechsler, and Bernd Becker

Abstract— **In this paper we present an Evolutionary Algorithm (EA) for *Detailed Routing Problems* (DRPs), like the *Channel Routing Problem* and the *Switchbox Routing Problem*. We combine EAs with domain specific knowledge, i.e. the genetic operators make use of the rip-up and reroute technique. The algorithm can work with two-layer and multi-layer problem instances.**

The efficiency of our algorithm is demonstrated by application to multi-layer channel routing benchmarks. Instances with up to five layers, 130 columns, and more than 60 nets are considered.

I. Introduction

Evolutionary Algorithms (EAs) have become more and more popular in the domain of search, optimization and machine learning [9], [2], [17]. In many areas they are superior to other classical optimization techniques, like e.g. gradient-descent. EAs have successfully been applied to general hard problems and recently are used for optimization in the area of *Computer Aided Design* (CAD), like placement, routing, test pattern generation and logic synthesis [15], [5], [3], [1], [10], [6], [4], [14]. Often it is useful to combine "pure" EAs, like genetic algorithms or evolutionary strategies, with problem specific knowledge [2], [17]. In our application a greedy algorithm is used to create the initial population and the genetic operators are established on problem specific methods.

One of the most important problems occuring during the physical design phase is the *Detailed Routing Problem* (DRP). A main task in the area of detailed routing is to connect pins of signals in a given routing region. Thereby some constraints must be considered, like the underlying routing model and the number of available layers. Furthermore, quality factors of the resulting layout must be noted, e.g. the number of tracks and vias or the net length which results from the wiring inside the channel.

In the last years various approaches based on EAs for detailed routing have been presented [19], [16], [18], [15], [8], [14]. Many EAs fail for large examples due to the runtimes and memory requirements. Recently, in [8] a new approach to the *Channel Routing Problem* (CRP) has been presented; a restricted version of the DRP. This approach makes use of problem specific knowledge, i.e. the genetic operators use a rip-up and reroute technique. In [8] a restricted model for detailed routing has been used: Only two layers were allowed and only channels were considered, i.e. the pins were only located at the lower and upper side of a rectangular region.

Institute of Computer Science, Albert-Ludwigs-University, 79110 Freiburg im Breisgau, Germany
email: {goeckel,drechsle,becker}@informatik.uni-freiburg.de

In this paper we present an extension of this approach, i.e. our algorithm can handle multi-layer channels. Additionally, it can deal with obstacles and switchboxes. We give experimental results that demonstrate the efficiency of our EA in comparison to previously published approaches.

The paper is structured as follows: In Section II the problem domain is described in detail and basic notations are given. Our EA approach is presented in Section III, where the construction of our problem specific operators and the algorithms are described. In Section IV the experiments are given and finally, the results are summarized in Section V.

II. Problem Domain

In this section we briefly describe the *Detailed Routing Problem* (DRP) under a set of constraints (see also [13]). Furthermore, basic definitions that are used in the following are introduced.

A. Definitions

A *switchbox* is a rectangular routing region (routing area) which is located between circuit blocks that are to be connected. (Notice, that due to the obstacles that are allowed also non rectangular structures can be considered.) The routing has to be performed on horizontal and vertical wires that are located in the switchbox. A horizontal row is called a *track* and a vertical wire is named *column*. The number of tracks defines the *width* of the switchbox. Furthermore, *pins* are allowed to be located on all side of the switchbox and represent the connection points on circuit blocks. A *net* is the union of pins that have to be connected and wire segments (tracks and columns) that are used to connect these pins.

The same definitions hold for a *channel* with the restriction that pins are only located at the upper and lower side and that the number of tracks in the channel is not fixed, i.e. in this case the CRP is to find a solution within a minmum number of tracks.

In our application the underlying detailed routing model is the *Manhattan Model* [13]. Therefore different nets have to be disjoint, i.e. two nets must not occupy the same wire segment. Several different interconnection layers are available for routing: We assume, that all pins are located on the lowest layer (*surface connection*). If a net is realized on different layers the connection between the two layers is called a *via*. Crosspoints of two different nets in the same layer are named *violations*.

Remark 1: Notice that a feasible routing solution has no violations.

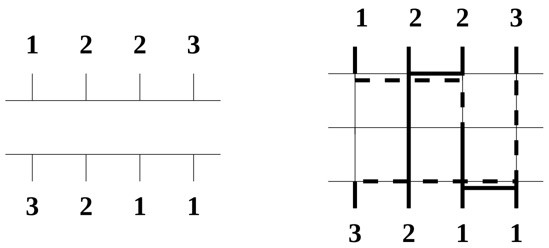

Fig. 1. Instance of two-layer CRP

Furthermore we need the definition of the *Minimum Manhattan Distance* [13]. It is defined as the minimum net length that is necessary to connect two pins (without respect to violations).

In our application we consider five optimization goals that are specified in the following (see also [11]):

Complete Routing Solution: The primary goal of our EA-based router is to find a valid routing solution, where each net is fully connected and no violations are present.

Width: The number of tracks which are needed to realize a routing solution has to be minimized to obtain a minimum routing area.

Net length: The net length of the routing solution is defined by the sum over the length of all nets. The length of a net largely influences the propagation delay of the realized circuit.

Number of vias: Each via directly requires costs in fabrication, forces a higher propagation delay and increases the defect probability in the resulting circuit.

Crosstalk: The crosstalk effect results from wire segments which are located on the same track segment in adjacent layers. Thus, we also minimize the number of wire segments which are located in parallel on adjacent layers.

In our approach the optimization goals are considered simultaneously, i.e. we perform multi-objective optimization. The importance of one of the quality factors in the optimization process is controlled by a weighted sum which is described in Subsection III-C.

Example 1: In Fig. 1 an instance of a two-layer routing channel (a special case of the DRP considered in this paper) is illustrated. The left picture shows the channel with pins that are to be connected. Pins belong to the same net if they are denoted by same integer numbers. The right picture shows a routing solution with two layers and channel width $w = 3$. Solid lines represent the connections in the lower layer and dashed lines represent the connections in the upper layer. Notice, that the routing solution is not minimum with respect to crosstalk effect, since two wire segments are located adjacent on the layers.

B. Lower Bound for the Number of Tracks

If the number of tracks in the routing area is not fixed the initialization algorithm of our EA starts with a lower bound for the track number. It is given in the following.

For the routing model considered in our application several layers are available and crosstalk should be avoided, but in general it is allowed.

Let I be an instance of the DRP and let $\{1, \ldots, m\}$ be the set of columns. Following [13] and [8] a lower bound for the number of tracks can be calculated by the following equation:

$$\widetilde{W}(I) = \max_i |\mathcal{N}_i| / \#layers, \tag{1}$$

where $i \in \{1, \ldots, m\}$ and $\mathcal{N}_i$ is the set of nets that have at least one pin in a column $\leq i$ and another in a column $\geq i$.

C. Rip-up and Reroute Technique

The genetic operators of our EA make use of domain specific knowledge and methods that are well-known in routing approaches. One of these is *rip-up and reroute* nets in a routing region.

Rip-up and Reroute: The heuristic starts from a completely wired solution, where each net is fully connected. The solution must not necessarily be feasible, i.e. violations may occur. Then one net η, $\eta \in \mathcal{N}$, is deleted ("rip-up") in the solution. The unconnected pins are connected using an arbitrary heuristic ("reroute"). (We make use of the random router RAND that is introduced in Subsection III-B.)

III. Evolutionary Algorithm

In this section we describe the *Evolutionary Algorithm* (EA) that is applied to the DRP.

The structure of our EA and the functionality of the genetic operators are discussed in detail. Finally, the parameter settings are given which are the same for all problem instances considered in this paper.

A. Representation

We use a multi-valued encoding to represent a solution of an instance of the DRP. The elements are given as finite strings which have no fixed length.

Our representation scheme works as follows: For each element we store the information which wire segments in the routing channel are occupied by net η, $\eta \in \mathcal{N}$, where $\mathcal{N}$ denotes the set of nets. The length of the strings is then limited by

$$C \cdot \sum_{\eta \in \mathcal{N}} length(\eta),$$

where C denotes a constant factor for handling the representation. Notice, that the length of the strings is limited by the sum over all nets inside the channel and not by the size of the channel. Therefore we are able to route large channels. (For more details see [8].)

A population is a finite set of elements defined above. To the elements various genetic operators are applied as described in Subsection III-D.

```
initialization_algorithm (region) {
 for each net do {
   {
     random_router (mode) ;
     if (violation) increment (mode) ;
     if (mode = max) random_router_violation (max) ;
   } while (net not routed) ;
 }
 return ;
}
```

Fig. 2. Sketch of the initialization algorithm

B. Initialization

In this subsection we describe the initialization algorithm INIT of our EA, that constructs the initial population of size $|\mathcal{P}|$. INIT consists of a heuristic that chooses the routes in the routing region. First the heuristic tries to choose the shortest connection between two pins. If this is impossible more and more detours are admitted. This heuristic is called *Random Router* and works as follows:

Random Router (RAND): Let $\{pin_1, \ldots, pin_m\}$ be a set of pins that have to be connected by the same net η. Begin with pin pin_1 and successively connect it to the other pins pin_i ($i \in \{2, \ldots, m\}$) by choosing one of (at most) six possible directions (these are: *left, right, forward, backward, up* and *down*) in the channel.
RAND works in several modes. Each mode in RAND depends on the choice of the probabilities for each direction. For the lowest mode M_0 in RAND the probability for the choice of the *shortest* direction in the channel is set to 1 and the remaining directions are set to 0. (Notice, that probability 1 for the shortest direction which goes straight ahead to the next pin and probability 0 for each other direction implies a heuristic to obtain a *minimized manhattan distance* (see also Subsection III-C).) The different modes M_i, $1 \leq i \leq max$, are adapted as follows: If i is increasing other directions than the shortest one become more and more likely. The number of different modes for RAND is constant (see Subsection III-F).

By using RAND the initialization algorithm INIT works as follows:

Initialization (INIT): Take the nets of the routing region in random ordering. Construct the first net by RAND in mode M_0. Then increment the mode of the algorithm RAND to M_1 and perform RAND in this new mode until a violation appears. In this case increase i, $1 \leq i \leq max$, and apply RAND again until M_{max} is obtained. If a legal routing solution without violations is not obtained perform the routing with RAND in mode M_{max} and allow the occurence of violations.

A sketch of the algorithm is given in Fig. 2.

To construct an initial population the algorithm INIT is performed $|\mathcal{P}|$-times with different random orderings of the nets.

C. Objective Function and Selection

The *objective function* calculates the fitness of each element in the population. Let x be an element of the population. First for each net η, $\eta \in \mathcal{N}$, which occurs in x a fitness value is determined by the function $F(\eta)$ given by:

$$
\begin{aligned}
F(\eta) = \Big(\; & a \cdot length(\eta) \\
& + b \cdot vias(\eta) \\
& + c \cdot violations(\eta) \\
& + d \cdot crosstalk(\eta) \\
\Big) \; & \\
/ \; & minimized\_manhattan\_distance(\eta)
\end{aligned} \tag{2}
$$

$length(\eta)$ denotes the net length of net η and $vias(\eta)$ ($violations(\eta)$) counts the number of vias (violations) of net η. $crosstalk(\eta)$ counts the number of parallel wire segments on adjacent layers. The weights of the sum are $a, b, c, d \in \mathbf{R}$ and $minimized\_manhattan\_distance(\eta)$ is an approximation of the *Minimum Steiner Tree* for net η, which is calculated by a simple greedy heuristic.

In our approach infeasible solutions may occur in the present population. Each violation of an element is counted by the penalty term $c \cdot violations(\eta)$. (More details of the choice of the factors a, b, c and d are given in Subsection III-F.) The fitness value $F(\eta)$ for each net η, $\eta \in \mathcal{N}$, is then divided by the minimized manhattan distance of the considered net; by this the fitness is normalized.

Using the formula described above the fitness of an individual x is then calculated by

$$fitness(x) = \sum_{\eta \in \mathcal{N}} F(\eta). \tag{3}$$

Thus, by evaluating the objective function we follow the optimization goals

1. to find a valid solution without violations and
2. to minimize the costs for the realization of the routing solution, i.e. the minimization of the net length, the number of vias, and the crosstalk effect.

During the EA run individuals are selected for creating offsprings by applying genetic operators (see Subsection III-D). In our approach the selection is performed by *roulette-wheel selection* [9], i.e. the elements are selected according to their fitness. Additionally, we make use of *steady-state reproduction* [2]: The worst elements in the population are deleted and replaced by newly created offsprings.

D. Genetic Operators

In the following the genetic operators are introduced. Each operator works on the encoded representation of the routing solutions and creates new elements. During the EA run violations are allowed, i.e. the operators need not necessarily create feasible solutions.

In our application we use two recombination operators for the creation of new individuals. Both methods create

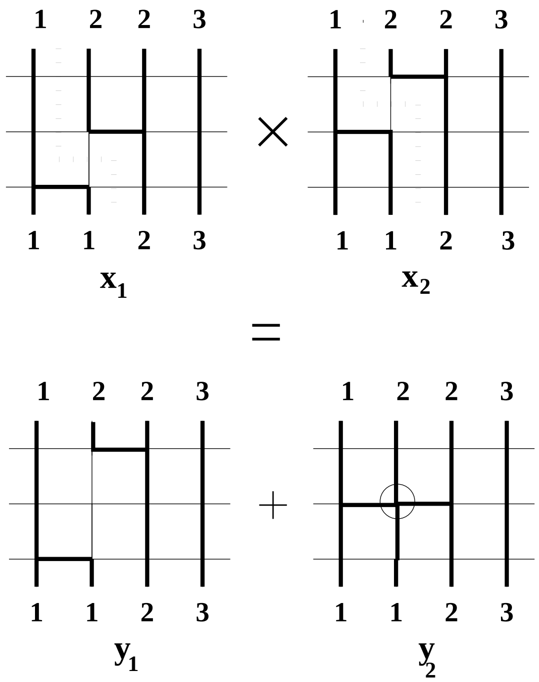

Fig. 3. Illustration of crossover

offsprings from two parents. The parent elements are selected as described in Subsection III-C.

The first crossover operator presented in this section needs an algorithm that calculates a cut position for crossing the elements:

CUT: Choose a net η, $\eta \in \mathcal{N}$, at random and set the cut position of the parent elements behind net η. This corresponds to an integer cut position d in the string representation. (Notice, that the crossover is applied to strings and therefore an integer value must be calculated as cut position.)

Then the two offsprings are constructed as follows:

Crossover: Create two new elements y_1 and y_2 from parents x_1 and x_2. Split each element x_1 and x_2 in two parts at a cut position d chosen by the function CUT. The first part of child y_1 (y_2) is taken from x_1 (x_2) and the second part of y_1 (y_2) is taken from x_2 (x_1).

For explanation we give an example where the effect of applying the crossover operator is illustrated.

Example 2: Fig. 3 shows an example of the crossover operator where only one layer is illustrated. The cut position is chosen behind net 1. Offspring y_1 has a shorter netlength than the parents and offspring y_2 is an invalid element.

Furthermore, we make use of a recombination operator that merges two parents to get one offspring:

Merge Crossover: Create one new element y from parents x_1 and x_2. y takes each net η, $\eta \in \mathcal{N}$ one after another from the parent, whose fitness $F(\eta)$ is of higher quality.

Additionally, we make use of two mutation operators that also work on the string representation. They are based on the rip-up and reroute technique described in Subsection II-C.

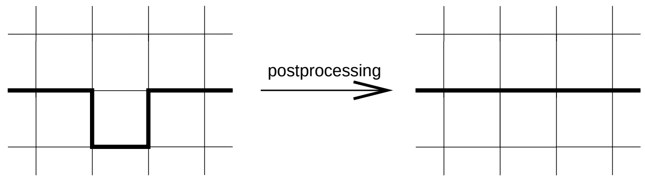

Fig. 4. Elimination of unnecessary detours

Mutation1 (MUT1): Create a new element by *rip-up and reroute* of one net randomly chosen.

Mutation2 (MUT2): Create a new element by performing MUT1 for two nets randomly chosen in parallel.

E. The Algorithm

Our EA computes feasible routing solutions for a given routing region and a fixed number of layers. Using the genetic operators it works as follows:

- The algorithm starts with a fixed number of tracks. It is calculated by formula (1).
- The initial population is generated by applying INIT.
- Each element in the initial population is evaluated using the objective function given by formula (3).
- In each iteration two parents are selected and one of the recombination operators is applied. Then one of the two mutation operators is selected with corresponding probability and applied to a randomly chosen element.
- The newly created offsprings are evaluated by the objective function.
- If no improvement is obtained for 5000 generations and a solution without violations is computed the algorithm stops. Otherwise, for each element one track is inserted into the channel. The elements are enlarged where the channel density is the highest (analoguosly to [15], [8]). Then the EA is restarted.
- Finally, the elements are optimized by a post processing. A local netlength optimization is then performed for the best element. For this detours are eliminated by one pass over the routing region. This method simplifies structures analoguosly to Fig. 4.

A sketch of the algorithm is given in Fig. 5.

F. Parameter Settings

In general the behaviour of EAs with respect to execution time and quality of the results largely depends on the choice of parameters. The number of modes for RAND in INIT is set to 6. (Thereby a compromise between runtime of the initialization algorithm and quality of the initial population is obtained.) The size of the population is set to 10, since for larger population sizes runtimes are no longer in an acceptable range.

TABLE I
COMPARISON: TWO-LAYER ROUTING/THREE-LAYER ROUTING

name	*columns*	*nets*	*2-layer*			*3-layer*		
			tracks	*net length*	*vias*	*tracks*	*net length*	*vias*
ex3a	89	44	10	1519	99	8	1418	162
ex3b	84	47	12	1602	131	9	1562	178
ex3c	103	54	12	2216	132	9	1809	181
ex4b	119	54	13	2268	184	9	1876	190
ex5	129	65	14	2575	222	10	2196	224

```
evolutionary_algorithm (region, # layers) {
 generate_initial_population () ;
 calculate_fitness () ;
 {
   apply_operators_with_corresponding_probabilities () ;
   calculate_fitness () ;
   update_population () ;
 } while (not terminal case) ;
 post_processing () ;
 return ;
}
```

Fig. 5. Sketch of the EA

Crossover is performed with a probability of 60%, *Merge Crossover* is applied with a probability of 10% and each mutation operator is applied with a probability of 20%.

For the objective function the factors a, b, c and d are chosen $a = 1$, $b = 10$, $c = 1000$ and $d = 10$. The choices of the weights are based on experiments. The primary optimization goal is to obtain feasible solutions during the run of the EA. For this, each violation of a solution is weighted by 1000. Both the number of vias and the crosstalk effect are weighted by 10. The net length is weighted by 1 only.

In the following we consider one fixed set of parameters. Notice, that in other applications different parameter settings might be of interest. Thus, the designer himself can influence the priorities of the optimization procedure.

IV. Experimental Results

We implemented our method described above in C++. All experiments have been carried out on a *Sun Sparc 20* workstation. In our experiments we consider channels from [12] with up to 130 columns and 65 nets. The runtimes for the examples considered range from several minutes (*ex3a* using two layers) up to one day (*ex5* using four layers) of CPU time.

In a first series of experiments we apply our EA to the given benchmarks. We consider channels with two layers and with three layers. The routing solutions for two layers are compared to the solutions resulting from three-layer routing. This should give the reader who is not familiar with channel routing an impression how many tracks can be saved by using one additional layer. As can be seen in Table I the number of tracks that have to be used can significantly be reduced if additional layers are available. Furthermore, using three layers the net length is decreased. This directly corresponds to an improved delay behaviour of the resulting circuit.

In our next experiment we consider realizations that allow four layers. To give an impression of the quality of the results we compare our method to the approach from [7] which is denoted by FFL in the following. In Table II it can be seen that our approach uses less tracks than FFL. Thus the routing area could be significantly reduced. The EA approach needs more execution time, but the results are of higher quality. But using tracks for routing can directly result in lower fabrication costs, if the number of layers is fixed in a given technology. Additionally, less net length is needed. Only the number of vias is increased[1].

Finally, we want to give an impression on the convergency behaviour of the optimization procedure using our objective function with parameter settings from Subsection III-F. In Fig. 6 the fitness values of the best element (with respect to the number of violations, vias, and net length) in a population are shown on the y-axis for channel *ex5* using four-layer realization. The x-axis gives the number of generations. (Notice, that due to the weights of the objective function the number of violations, vias, and net length given on the y-axis are not equal to the corresponding fitness.) It can be seen that first the number of violations, which are mainly penalized in our objective function, are reduced until the best element has no further violation. Then, the remaining attributes of a routing solution, like the number of vias and the net length are optimized.

V. Final Remarks

We presented an *Evolutionary Algorithm* for multi-layer detailed routing. The algorithm can be applied to channel routing and switchbox routing. Experiments have demonstrated the quality of the results.

It is focus of future work to improve the considered model with respect to routing of *Multi Chip Modules* (MCMs). There the routing region can be modeled by channels,

[1] In [7] a slightly simpler routing model has been considered, i.e. *surface connection* was not used. Thus, the number of vias obviously could be decreased.

TABLE II
FOUR-LAYER ROUTING

name	*columns*	*nets*	FFL			EA		
			tracks	*net length*	*vias*	*tracks*	*net length*	*vias*
ex3a	89	44	8	868	68	6	1352	186
ex3b	84	47	9	1363	107	7	1457	198
ex3c	103	54	9	1727	125	7	1871	254
ex4b	119	54	9	2205	179	8	2031	257
ex5	129	65	10	2200	150	7	2022	263

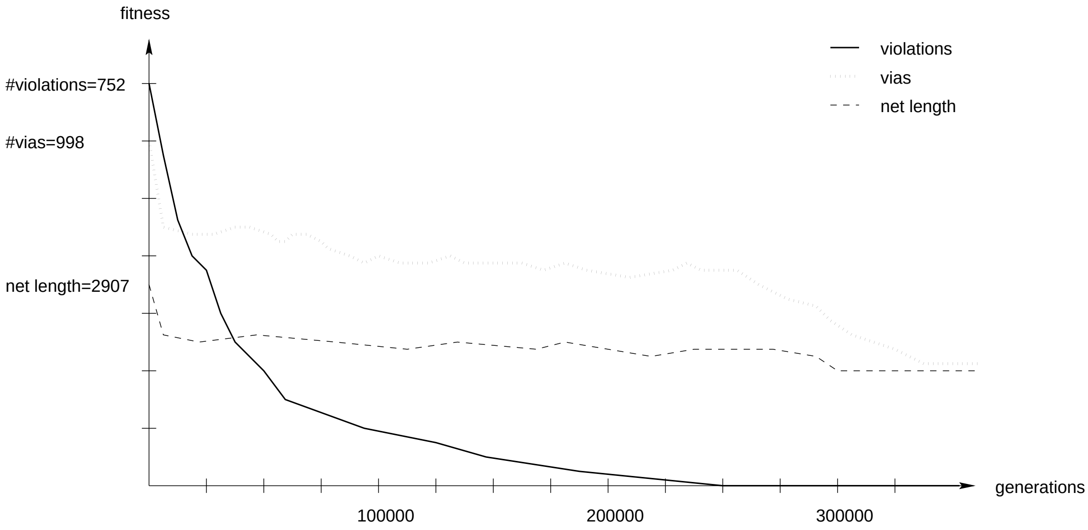

Fig. 6. Behaviour of convergency of example *ex5*

switchboxes, and obstacles. One main goal is then to find feasible routing solutions using a minimum number of layers.

ACKNOWLEDGEMENT

The authors like to thank Gregor Pudelko for his help with the implementation of the methods presented in this paper.

REFERENCES

[1] F. Corno, P. Prinetto, M. Rebaudengo, and M.S. Reorda. Comparing topological, symbolic and ga-based atpgs: an experimental approach. In *Int'l Test Conf.*, pages 39–47, 1996.

[2] L. Davis. *Handbook of Genetic Algorithms.* van Nostrand Reinhold, New York, 1991.

[3] R. Drechsler, B. Becker, and N. Göckel. A genetic algorithm for minimization of Fixed Polarity Reed-Muller expressions. In *Int'l Conf. on Artificial Neural Networks and Genetic Algorithms*, pages 392–395, 1995.

[4] R. Drechsler, N. Göckel, and B. Becker. Learning heuristics for OBDD minimization by evolutionary algorithms. In *Parallel Problem Solving from Nature, LNCS 1141*, pages 730–739, 1996.

[5] H. Esbensen. A macro-cell global router based on two genetic algorithms. In *European Design Automation Conf.*, pages 428–433, 1994.

[6] H. Esbensen and E.S. Kuh. EXPLORER: an interactive floorplaner for design space exploration. In *European Design Automation Conf.*, 1996.

[7] S.C. Fang, W.S. Feng, and S.L. Lee. A new efficient approach to multilayer channel routing problem. In *Design Automation Conf.*, pages 579–584, 1992.

[8] N. Göckel, G. Pudelko, R. Drechsler, and B. Becker. A hybrid genetic algorithm for the channel routing problem. In *Int'l Symp. Circ. and Systems*, pages IV:675–IV:678, 1996.

[9] D.E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addision-Wesley Publisher Company, Inc., 1989.

[10] M.S. Hsiao, E.M. Rudnick, and J.H. Patel. Alternating strategies for sequential circuit atpg. In *European Design & Test Conf.*, pages 368–374, 1996.

[11] R. Joobbani. *An Artificial Intelligence Approach to VLSI Routing.* MA: Kluwer Academic Publishers, 1986.

[12] B.W. Kernighan. An optimum channel routing algorithm for polycell layouts of integrated circuits. In *Proceedings of the 10th Design Automation Workshop*, pages 50–59, 1973.

[13] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* Teubner, Wiley, 1990.

[14] J. Lienig. Channel and switchbox routing with minimizaed crosstalk - a parallel genetic algorithm approach. In *VLSI Design Conf.*, pages 27–31, 1997.

[15] J. Lienig and K. Thulasiraman. A genetic algorithm for channel routing in VLSI circuits. *Evolutionary Computation*, 1(4):293–311, 1993.

[16] Y.-L. Lin, Y.-C. Hsu, and F.-S. Tsai. Silk: A simulated evolution router. *IEEE Trans. on CAD*, 8(10):1108–1114, 1989.

[17] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1994.

[18] A.T. Rahmani and N. Ono. A genetic algorithm for the channel routing problem. In *Proceedings of ICGA*, pages 494–498, 1993.

[19] H. Shin and A. Sangiovanni-Vincentelli. A detailed router based on incremental routing modifications: Mighty. *IEEE Trans. on CAD*, 6(6):942–955, 1987.