International Symposium on Circuits and Systems (ISCAS)
pages 675-678, Atlanta, May 1996

A HYBRID GENETIC ALGORITHM FOR THE CHANNEL ROUTING PROBLEM

Nicole Göckel[1] *Gregor Pudelko*[2] *Rolf Drechsler*[1] *Bernd Becker*[1]

[1] Institute of Computer Science, Albert-Ludwigs-University, 79110 Freiburg im Breisgau, Germany
email: goeckel@informatik.uni-freiburg.de
[2] Department of Computer Science, Johann Wolfgang Goethe-University, 60054 Frankfurt am Main, Germany

ABSTRACT

We present a *Hybrid Genetic Algorithm* (HGA) for the *Channel Routing Problem* (CRP). To do so we combine a *Genetic Algorithm* (GA) with domain specific knowledge, i.e. the genetic operators make use of the rip-up and reroute technique. Thereby the execution time of our method is faster than previously presented evolutionary based approaches. Furthermore, concerning space complexity we show by experiments that our HGA can handle large channels (with more than 100 columns).

1. INTRODUCTION

During the last years *Genetic Algorithms* (GAs) have become more and more popular in the domain of search, optimization and machine learning [6, 1]. In many areas they are superior to other classical optimization techniques, like e.g. gradient-descent.

GAs have succesfully been applied to general hard problems [2] and recently are used for optimization in the area of *Computer Aided Design* (CAD), like placement, routing test pattern generation and logic synthesis (see e.g. [4, 3]).

Often it is useful to combine GAs with problem specific knowledge [1, 12]. The resulting GAs are called *Hybrid GAs* (HGAs) and their genetic operators are established on problem specific methods.

The *Channel Routing Problem* (CRP) is a special type of detailed routing in the VLSI design process. One of the main tasks in the area of detailed routing is to connect pins of signals in a rectangular 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.

The CRP is NP-complete [15] and thus there is no hope to find a deterministic polynomial algorithm to solve the CRP. Therefore several heuristical algorithms have been proposed (see e.g. [7, 8]).

In the last years various routing approaches based on evolutionary strategies have been presented [14, 11, 13, 10]. Many GAs for the CRP fail for large examples due to the large runtimes.

In our application we developed a HGA for the CRP where problem specific knowledge is incorporated. We start with experiments for small benchmarks and compare our method to an approach from [10] to verify the quality of our results. It is shown by experiments that the runtimes are improved, if problem specific heuristics are used.

Finally, we also applied our HGA to large benchmarks that are used in industrial applications.

2. PROBLEM DOMAIN

In this section we briefly describe the *Channel Routing Problem* (CRP) under a set of constraints as e.g. defined in [9].

2.1. Definitions

A *channel* is a rectangular routing region between circuit blocks that are to be connected. The routing has to be performed on horizontal and vertical wires that are located in the channel. A horizontal row is called a *track*. The number of tracks defines the *width* of the channel. Furthermore, *pins* are located at the top and the bottom side of the channel and represent the connection points on circuit blocks. A *net* is the union of the pins that have to be connected and the wire segments (tracks and columns) that are used to connect these pins.

In our application the underlying detailed routing model is the *Manhattan Model* [9]. Two different interconnection layers are available for routing: the upper layer and the lower layer[1]. We assume, that all pins are located on the lower layer. 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*. Notice, that a valid routing solution has no violations. Furthermore we need the definition of the *Minimum Manhattan Distance*. It is defined as the minimum net length that is neccessary to connect two pins (without respect to violations).

In our application we consider four optimization goals that are specified in the following (see also [7]):

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

Channel 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, means a higher propagation delay and increases the defect probability in the resulting circuit.

[1] We make use of *unrestricted overlapping*, i.e. we use the same routing model as [7, 11, 10]. Therefore the track assignments of nets on the upper and lower layer do not need to be disjoint.

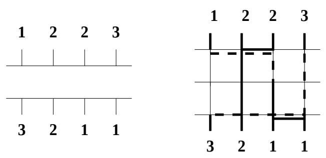

Figure 1. Instance of the CRP

In our approach the optimization goals are considered simultaneously. The importance of one of the quality factors in the optimization process is controlled by a weighted sum which is described in Subsection 3.3.

Example 1 In Figure 1 an instance of a routing channel is illustrated. The leftmost picture shows the channel with the pins that are to be connected. Pins that are denoted by the same integer belong to the same net. 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.

2.2. A Lower Bound for the Channel Width

Our HGA makes use of some further properties that are described next.

A lower bound for the number of tracks is proven in [9] for the CRP in the *Manhattan Model* with restricted overlapping and can be calculated by equation (1). Let I be an instance of the CRP and let $\{1, \ldots, m\}$ be the set of columns. Then a lower bound for the channel width is given by

$$W(I) = \max_i |\mathcal{N}_i|, \qquad (1)$$

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 $\geq i$.

For the routing model considered in our application two layers are available and overlapping is allowed. As can easily be seen a lower bound can then be calculated by the following equation:

$$\widetilde{W}(I) = \max_i |\mathcal{N}_i|/2 \qquad (2)$$

3. HYBRID GENETIC ALGORITHM

In this section we describe the *Hybrid Genetic Algorithm* (HGA) that is applied to the CRP.

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

3.1. Representation

We use a multi-valued encoding to represent a solution of an instance of the CRP. The elements are represented 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, $1 \leq \eta \leq |\mathcal{N}|$, 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 [5].)

A population is a finite set of elements defined above. To the elements various genetic operators are applied as described in Subsection 3.4.

3.2. Initialization

In this subsection we describe the initialization algorithm INIT of our HGA, that constructs the initial population. INIT consists of a heuristic that chooses the routes in the channel. 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 described in the following:

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 the five possible directions (these are: *left, right, forward, backward* and *layer change*[2]) 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 greedy algorithm.) The different modes M_i, $1 \leq i \leq max$, are adapted as follows: If M_i is increasing other directions than the shortest one become more and more likely. The number of modes for RAND is constant and considered in Subsection 3.6.

By using RAND the initialization algorithm INIT is described:

Initialization (INIT): Take the nets of the channel in an ordering which is determined at random. 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 appearence of violations.

3.3. 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 objective function

$$F(\eta) = \frac{a \cdot length(\eta) + b \cdot vias(\eta) + c \cdot violations(\eta)}{minimized\_manhattan\_distance(\eta)}. \qquad (3)$$

$length(\eta)$ denotes the net length of net η and $vias(\eta)$ ($violations(\eta)$) counts the number of vias (violations) of net η. The weights of the sum are $a, b, c \in \mathbf{R}$ and

[2] In an extension to multilayer routing six directions have to be considered.

minimized_manhattan_distance(η) is an approximation of the *Minimum Steiner Tree* for net η.

In our approach non valid solutions may occur in the present population. Each violation of an element is counted by the penalty term $c{\cdot}violations$. (More details of the choice of the factors a, b and c are given in Subsection 3.6.) 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 formula (3) the fitness of an element x is then calculated by

$$fitness(x) = \sum_{\eta \in \mathcal{N}} F(\eta). \qquad (4)$$

Thus, by evaluating the objective function we pursue the following 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 netlength and the number of vias.

During the GA run elements are selected for creating new elements by applying genetic operators (see Subsection 3.4.). In our approach the selection is performed by *roulette wheel selection* [6], i.e. the elements are selected according to their fitness. Additionally, we make use of *steady state reproduction* [1]: The worst elements in the population are deleted and replaced by newly created offsprings.

3.4. Genetic Operators

Now the genetic operators are introduced. Each operator works on the encoded representation of the routing solutions and creates new elements where violations are allowed.

In our application we use a crossover operator for the recombination of elements. This method creates two offsprings from two parents. The crossover operator needs an algorithm that calculates a cut position for crossing the elements. This algorithm is called CUT and specified in the following:

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 a 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.)

For crossover the parents are selected as described in Subsection 3.3. and two offsprings are constructed as follows:

Crossover: Create two new elements y_1 and y_2 from two parent elements 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 from child y_1 (y_2) is taken from x_1 (x_2) and the second part from 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 in the routing channel is illustrated.

Example 2 Figure 2 shows an example of the crossover operator where only one layer is figured. 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.

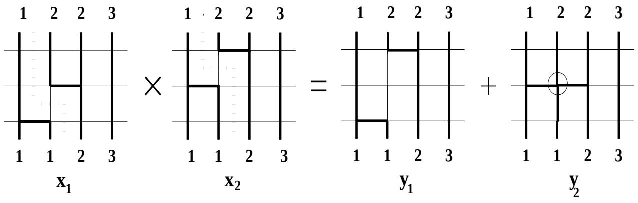

Figure 2. Illustration of crossover

```
genetic_algorithm (channel) {
  generate_initial_population () ;
  calculate_fitness () ;
  {
    apply_operators_with_corresponding_probabilities () ;
    calculate_fitness () ;
    update_population () ;
  } while (not terminal case) ;
  post_processing () ;
  return ;
}
```

Figure 3. Sketch of the HGA

Additionally, we make use of two mutation operators that work also on the string representation. They are based on the rip-up and reroute technique which is well-known in routing approaches.

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

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

3.5. The Algorithm

Using the genetic operators our HGA works as follows:

- The lower bound for the number of tracks $\widetilde{W}(I)$ for the elements is calculated by formula (2).
- The initial population is generated by applying INIT. The size of the population is given in Subsection 3.6.
- In each iteration two parents are selected by the method from Subsection 3.3. and then the crossover operator is applied. Then one of the two mutation operators is selected with corresponding probability and applied to one randomly chosen element.
- 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 [10]). Then the HGA is restarted.
- Finally the elements are optimized by a post processing. A local netlength optimization is then performed for the best element. (For more details see [5].)

A sketch of the algorithm is given in Figure 3.

3.6. Parameter Settings

In general the behavior of HGAs with respect to the execution time and the quality of the results largely depends on the choice of the parameters (see e.g. [1]).

Table 1. Results for small channels

name	*columns*		HGA	LT
		rows	6	6
Jo6_16	11	*net length*	115	116
		vias	16	15
		time	43.1 min	48.9 min
		rows	4	4
Bursteins	12	*net length*	82	82
Difficult		*vias*	8	8
Channel		*time*	0.5 min	9.6 min

Table 2. Results for larger channels

	columns		HGA	HGA*
		rows	4	4
fdc01	40	*vias*	18	16
		time	3.3 min	10.3 min
		rows	50	50
lprim1s	149	*vias*	104	117
		time	45.2 min	244.2 min

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.) In our application the quality of an initial population influences the convergency of the HGA (see Section 4.). The size of the population is set to 10 if the number of pins which have to be connected is smaller than 100. For larger examples the population size is set to 15. Crossover is performed with a probability of 60% and each mutation operator is applied with a probability of 20%. The factors a, b and c from the objective function are chosen $a = b = 1$ and $c = 1000$. This is done to obtain valid solutions during the run of the HGA as fast as possible.

4. EXPERIMENTAL RESULTS

We implemented our method described above in C++. All experiments have been carried out on an *HP 9000/750* workstation. All runtimes are given in CPU seconds.

In a first series of experiments we consider the routing solutions for small channels up to 18 columns. We compare our results to [10] where a comparison to other routing approaches is given. As can easily be seen in Table 1 the quality of our routing solutions (column HGA) is not worse than in [10] (column LT), but the runtime is improved.

To give an impression of the quality of our HGA and especially of the initialization algorithm INIT we applied our HGA to larger examples that are given in Table 2. *Iprim1s* is used in an industrial application and *fdc01* is a channel that has been constructed for our application. Column HGA shows the results obtained by the method described above. HGA* denotes a HGA where the initial population is obtained at random, e.g. without using INIT. As can be seen in Table 2 the application of the heuristic during the initialization (HGA) directly improves the execution time and the quality of the results are nearly the same. Therefore our HGA is also able to route industrial benchmarks up to 700 columns. The execution time that is needed for these large examples is about 24 CPU hours.

Finally, we want to mention that further results and their discussion are given in [5].

5. FINAL REMARKS

We presented a *Hybrid Genetic Algorithm* for the *Channel Routing Problem*. The results presented above are as good as the routing solution from [10] and additionally the execution times are improved while using a heuristical initialization algorithm. Furthermore, our method can handle larger benchmarks (within reasonable time bounds) than other previously presented approaches that are based on evolutionary strategies.

It is focus of current work to improve the present model with respect to several points: restricted overlapping, multilayer routing and further routing geometries, e.g. switchboxes.

ACKNOWLEDGEMENT

The authors like to thank Ines Peters and Paul Molitor for their helpful discussions.

REFERENCES

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

[2] K.A. DeJong and W.M. Spears. Using genetic algorithms to solve NP-complete problems. In *Proceedings of ICGA*, pages 124–132, 1989.

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

[5] N. Göckel, G. Pudelko, R. Drechsler, and B. Becker. A hybrid genetic algorithm for the channel routing problem. Technical report, Albert-Ludwigs-University, Freiburg, 1996.

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

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

[8] A. LaPaugh and R.Y. Pinter. Channel routing for integrated circuits. *Annual Reviews*, pages 307–367, 1990.

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

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

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

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

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

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

[15] T.G. Szymanski. Dogleg channel routing is NP-complete. *IEEE Trans. on CAD*, 4(1):31–41, 1985.