EDUCATING INITIAL SOLUTIONS FOR GENETIC ALGORITHMS: A CHIP PLANNING OPTIMIZATION EXAMPLE

O. Peyran *W. Zhuang*

Institute of High Performance Computing,
1 Science Park Road, #01-01 The Capricorn
Singapore 117528, Singapore

ABSTRACT

This paper aims at addressing a particular aspect of multi-objective optimization problems in Electronic Design Automation (EDA). More specifically, we will present our study of the impact on quality and performance of evolutionary algorithms initial solutions. A new scheme for initial population generation is presented, which improves the quality and efficiency of chip planning optimization using genetic algorithm. The underlying concept is to educate the initial solutions by artificially introducing high quality genes that will only show up in the natural evolutionary process. Since these genes are initially hidden, they do not impose their strength in the early phase of evolution, therefore preventing any premature convergence.

1. INTRODUCTION

The increasing density in semiconductor technologies has brought new difficulties in the design automation area. The EDA industry has moved into the multi-dimension multi-objective system optimization era, where tools are performing optimization designers cannot handle anymore. Combining different objectives in an optimization problem may however lead to more than increasing complexity. This is particularly true with EDA problems, where optimization criteria are often correlated and conflicting (for instance minimizing wire length and reducing routing congestion).

Multi-objective evolutionary algorithms (MOEAs) have become very popular over the last decade to solve such problems. Many techniques have been presented in the literature ([1][2] for reviews). A first discriminator among them is the decision method used to sort and select solutions. The first approach consists in combining all the criteria into a single fitness function. If this approach is easily implemented, it has the drawback of only focusing on a portion of the optimal set of solutions. An alternative is to use several fitness functions and to rank the solutions using the concept of pareto dominance: solution x dominates solution y if none of x's fitness values is worst than y's and if at least one of them is better. The objective is then to find the best pareto-optimal set (set of solutions that are not dominated by any other).

Pareto-based selection techniques are currently the most popular choices in MOEAs. If they all share the same objective, that is reaching the true-Pareto optimal set in the shortest time, they differ by a multitude of variations. One can cite [3][4] for the ranking methods.

Several studies have been presented regarding the impact on efficiency of population diversity. In [5] Hanne introduced the efficiency preservation (resp. negative efficiency preservation) properties expressing that the dominating set is reduced (resp. increased) during the evolution and studied the different schemes. Osyczka and Krenich [6] proposed a filtration method in which less important Pareto optimal solutions are removed from the existing set in order to reduce the computation time.

Even though the quality of initial solutions can have a strong impact on efficiency, little research exists concerning the choice of initial population. Initial populations are usually randomly generated in order to provide unbiased starting point. Indeed, good quality initial solutions may lead to premature convergence to local optima for one given optimization criterion. An alternative is to introduce seeds of good solutions into a randomly generated population. In [7] Valenzuela and Smith showed that seeding could achieve substantial improvements. On the other hand, it did not have any impact for some of the problems considered in [8]. A reason could be that, though these solutions are of good quality, there is no insurance that their genes will be transmitted to the rest of the population, since the quality genes are present only in very few initial solutions. In [9], Hill modified the random generation method of the whole population in order to force the generation of feasible solutions for the knapsack problem. This approach did not have an impact on the quality of the final result, since no the initial solution quality was not altered.

Our approach is to prepare all the initial solutions by

artificially introducing good quality genes for a given optimization criterion. The fitness of the initial solution is not noticeably affected, therefore preventing from converging prematurely, but the hidden genes provide the background for the future generation of high quality solutions. These "education" of initial solutions is demonstrated with a chip planning optimization problem.

The remainder of this paper is organized as follows: in the next section, the chip planning problem is defined; in section 3 we demonstrate our approach of initial solution generation and present an algorithm for the chip-planning problem; section 4 shows experimental results.

2. THE CHIP PLANNING PROBLEM

Chip planning is the task of optimizing the top-level floorplan of block-based designs. The need for such an optimization comes from the long interconnection delays between physical blocks that may have a critical impact on timing closure and routing congestion.

Chip planning can be considered at very early design phases when only the system specification is known. At this stage, the design is defined as a set of interconnected blocks. These blocks can be either hard intellectual property blocks (IPs) or "soft blocks". Hard IPs have fixed physical implementations (block shape and pin assignment). Soft blocks are functional blocks that have not been synthesized yet. Though it is possible, based on experience, to evaluate the area of such blocks, their shape and pin assignment are flexible.

The chip-planning problem is formulated as follows:

- *Given a set of hard or soft interconnected blocks*
- *Find the best shape and best pin assignment for soft blocks and the best position for all blocks*
- *In order to optimize criteria such as*
    - *Timing closure (interconnection delays)*
    - *Area*
    - *Top-level routing congestion*
    - *Chip aspect ratio*

Many solutions to this problem have been proposed. They can be segregated between local [10] and global optimizations [11][12][13]. The local approach is usually based on constructive heuristics. These heuristics are not very suitable for multi-objective optimizations since the quality of results decreases rapidly with the problem size.

For global multi-objective optimization, genetic algorithms are quite effective since they simultaneously deal with a population of possible solution. Moreover, their stochastic aspect makes them a good candidate to solve problems with intricating optimization criteria.

The main drawback of global optimizations is the long computation time. In the following section, we present a method to educate GA initial population in order to improve both quality and performance.

3. EDUCATING THE INITIAL SOLUTIONS

The theoretical foundations of GA rely on the concept of schema - a template allowing exploration of similarities among solutions. Genetic algorithms converge to near-optimal results through the juxtaposition of short, low-order, high-performance schemata [14].

Our approach is to alter the random generation of initial solutions by artificially introducing high-performance schemata. The generated solutions are not so much expected to be of better quality compared to randomly generated ones, but, more importantly, they are more fitted for the optimization process. In other words, instead of providing some selected experts in one criterion – seeding – or a wide diversity of randomly chosen candidates, we educate the initial solutions in order to prepare them for the following optimization.

We have evaluated our approach by generating initial solutions of a multi-objective chip-planning problem that are trained for delay optimization. These initial solutions are then globally optimized by a genetic algorithm for area, maximum interconnection delay, routing congestion and chip aspect ratio. The initial solutions differ from randomly generated ones in the sense that highly interconnected blocks are kept close to each other. The idea is that the final solutions will show the same characteristic (schema), leading to good delays. The random aspect is still dominant in order to ensure a wide diversity of solutions.

3.1 Genetic Algorithm for chip planning optimization

In this paper, we will focus on the results for area and delay optimization. Before presenting the algorithm to generate initial solutions, we introduce the format and metrics used in the GA.

A solution to the problem is defined by the position of each block, their aspect ratio and pin assignment. The evaluation of the delay is done using critical-sink Steiner trees for Elmore delay optimization [15]. The metric for area evaluation is the area of the smallest rectangle bounding all the blocks, after compaction. The ratio for each block is chosen randomly. The characteristics of the GA operators will not be discussed in this paper.

The binary representation of the block placement is done using a reverse polish expression (RPE) representing slicing structures. A slicing structure is a placement that can be recursively vertically or horizontally partitioned [16]. Binary trees are used to represent slicing structures, with $V=\{1,2,...,n, +, *\}$ being the set of vertices, where $\{1,..., n\}$ is the set of leaves and represent the indexes of the n blocks while + and * respectively represent a horizontal or vertical cut between two sub-trees. A RPE can be obtained by a post-order traversal of the slicing tree. A RPE is made up of n blocks

and *n-1* operators. When reading the RPE from left to right, at any position we have $n \geq nBlo \geq nOp + 1$, with *nOp* the number of operators encountered (+ or *) and *nBlo* the number of blocks. Fig. 1 shows an example of slicing structure, slicing tree and RPE.

The random generation for the initial structures is made as follows (*G[i]* is the element of the RPE of index *i* when reading from left to right; *random()* is a function returning the value OP with a probability of 0.5):

```
nOp=nBlo=0
For i=0 to 2*n-2 do
 If (n>nBlo>nOp+1)
  If random() = OP
   G[i] = random choice of operator
   nOp++
  Else
   G[i] = random choice of block among
   non-selected ones
   nBlo++
 Else If n = nBlo
  G[i] = random choice of operator
  nOp++
 Else If nBlo = nOp+1
  G[i] = random choice of block among
  non-selected ones
  nBlo++
```

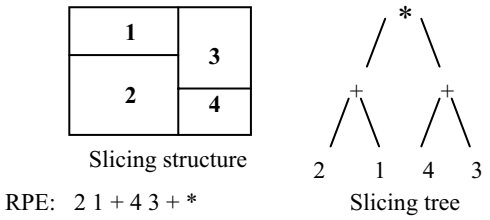

Fig. 1: Slicing structure, slicing tree and RPE

3.2 Introducing schemata into initial solutions

We propose an education system based on the previous random generation algorithm. It relies on a connectivity matrix that evaluates the block interconnectivity based on the path definitions.

A path is an alternating sequence of blocks and pins, with each pin belonging to the block immediately preceding it and being connected to a pin of the block succeeding it. From one path, we generate the complete graph *G* consisting of all the blocks in the path. From *P*, the set of all the complete graphs generated by all the paths, we define the interconnection matrix as follows:

$$M_{i,j} = \left| \{ e_{i,j}^p ; \exists G_p = (V_p, E_p) \in P / e_{i,j}^p = v_i \rightarrow v_j \in E_p \} \right|$$

In other words, M_{ij} represents the number of times that blocks *i* and *j* are in the same path.

The education algorithm modifies the "*random choice of block among non-selected ones*". Instead of a random choice, if a new block has to be selected, the choice is "the block mostly-connected to the already chosen blocks".

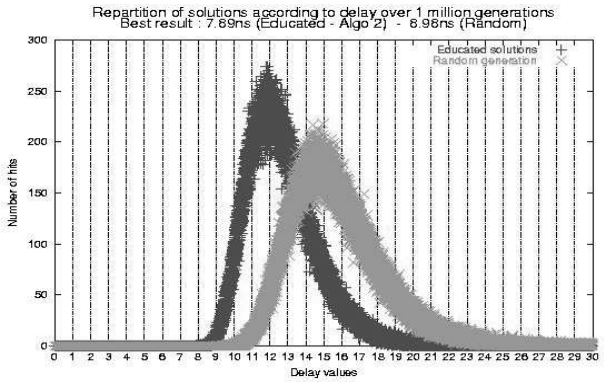

Fig. 2: Solution repartition according to delay

We call "block mostly-connected to a set of blocks V_{prev}" the block *j*, among all the not previously selected blocks, that maximizes $\Sigma_i M_{ij}$, with $i \in V_{prev}$.

Moreover, instead of "filling" the RPE from left to right, we fill it from either its left or right end. The first block is the most connected block, *i.e.* the block *j* that maximizes $\Sigma_i M_{ij}$, with $i \in [1..n]$. Then the next position to be filled will be either to the left or to the right of the first block. The choice to fill the left or right position is made randomly.

The block selection considers the most connected block to the, at most, P consecutive blocks preceding the new position (if we have chosen the right position) or following it (if we have chosen the left position). The selection is repeated until all the positions are filled.

Using a benchmark with 49 blocks (ami49), a statistical analysis was performed by generating one million initial solutions randomly and using the education system. The results are shown on Fig. 2.

As expected, the randomly generated solutions have poor delay and very poor area. The educated solutions show on average a better delay, though the best-generated delay (7.89ns) is far from the best known result after optimization (6.75ns). The area and congestion are, on average, equivalent to the random generation.

4. OPTIMIZING EDUCATED SOLUTIONS

We have presented in the previous section an algorithm to generate initial solutions for a chip planning optimization problem. This section presents the quality and performance of the GA using educated vs. randomly generated initial solutions.

4.1 Test methodology

The fitness function of our genetic algorithm that evaluates the quality of a given solution is based on a ranking among the various optimization criteria. The ranking is dynamic and takes into consideration the target

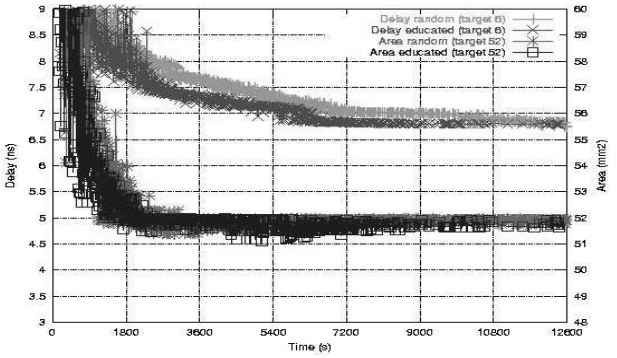

Fig. 3: Educated vs. Random. Area target 52

value for each criterion. The benchmark used is a 49-block design (ami49) with only flexible blocks (therefore the block ratios have to be optimized as well). Three target values were considered for area: *52*, for easy area optimization; *48.86*, for reasonably difficult area optimization (the value is 10% over the lower bound of the chip area); *0*, for intensive area optimization. Each test was run ten times in order to get a statistical approach.

4.2 Experimental results

Fig. 3, 4 and 5 show the experimental results for the different area targets. Each figure represents the best results over the ten different runs (using the same parameters). A point in the graph depicts a new non-dominated solution. The horizontal axis indicates the time when the solution is generated.

4.2.1 Area target 52: focus on delay

The target for area is reached after roughly twenty minutes, as shown in Fig. 3. The curve for area is similar for random or educated initial solutions. Once the area target is met, the GA focuses on delay optimization. After ninety minutes, the GA using educated solutions stabilizes to around 6.8ns. At the same time the GA using random solutions has generated results with 5% higher delay. After a while, since all the optimization is now focus on the delay, the random solution based GA converged to the same value as the educated one.

This test shows that when the optimization effort is focused primarily on the criterion for which the solutions have been trained, educated initial solutions brings an advantage in terms of performance. The final result could be reached in half of the computation time needed by the random initial solution approach.

4.2.2 Area target 48.86: medium area optimization

Fig. 4 shows the result of an optimization with area target 10% over the lower bound of the chip area. We can see that when two criteria have to be optimized concurrently, educated solutions make a difference.

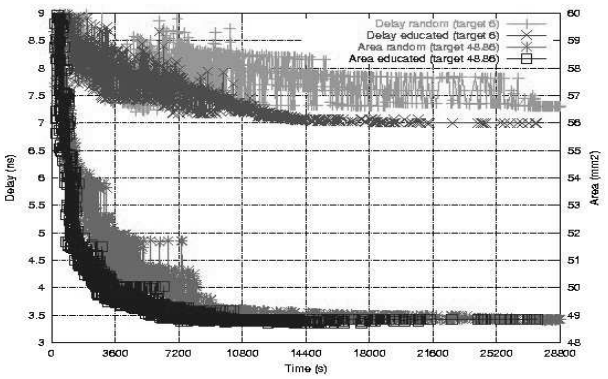

Fig. 4: Educated vs. Random. Area target 48.86

After 4 hours, the GA with educated initial solutions converges to a result satisfying the area constraint with a delay of 7ns. The GA using random initial solutions only converges after eight hours for a delay 5% higher. We actually ran the optimization for random solutions for fourteen hours. In the end, the best delay is still 3% higher compared to educated solutions.

In other words, thanks to educated initial solutions, we manage to reach a quality of result that random initial solution based GA cannot reach even using three times more computational time.

4.2.3 Area target 0: intensive area optimization

In this test, the optimization focus is on area though the genetic algorithm still needs to optimize the delay. Initial solutions that have been educated for delay are still making a difference. Indeed, the initial preparation for delay gives more computational resources for area optimization. As a result, random and educated based GA both converge to the same value for delay (Fig. 5). However, the educated based GA gives much better area compared to the random one.

Considering the trend of area quality improvement for the random-based GA, it would take, using the traditional approach, a prohibitively long computation time to reach the same quality as the one attained by the educated-based GA in four hours.

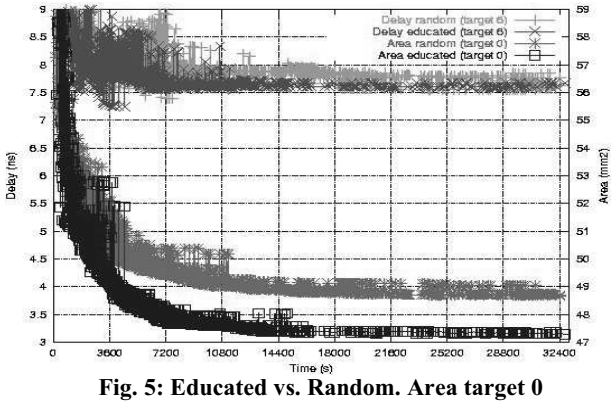

Fig. 5: Educated vs. Random. Area target 0

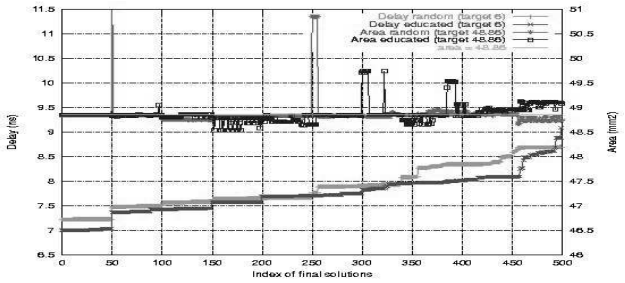

Fig. 6: Results for area target 48.86 over 10 runs

4.2.4 Statistical results

The previous figures showed the best performance over time among ten different runs. Fig. 6 shows all the final results for area target 48.86 after 8 hours. The fifty best solutions of the ten runs are represented consecutively.

One can see that nine times out of ten, educated solutions improve the overall quality of result. Only one run gave similar results for random and educated initial solutions. The best result (first fifty indexes) is the final result of the optimization represented in Fig.4.

For a given set of parameters, we also studied the importance of the quality of the initial educated solutions. It appears that the best results are not necessarily achieved with the educated solutions of highest quality. In other words, the impact of educated solutions is not due to their initial fitness (*i.e.* the initial value for the various optimization criteria). This observation confirms that the fitness of initial solutions is not important and that the aptitude to improve lies in the hidden high-performance schemata that have been artificially introduced.

5. CONCLUSION

We have presented a methodology to improve the quality and performance of multi-objective genetic algorithms. The example of a genetic algorithm for the chip planning optimization problem was studied.

Our methodology consists of preparing the initial solutions on which the optimization will operate. Instead of generating high quality initial solutions, which generally leads to premature convergence, we artificially introduce high performance schemata in randomly generated solutions. In our case, these solutions were "educated" to generate solutions of good delay by putting highly connected blocks close to each other.

Extensive experimental results showed that our approach substantially improves both quality of result and performance.

6. REFERENCES

[1] C. A. Coello Coello, "A short tutorial on Evolutionary Mutliobjective Optimization", *EMO 2001, LNCS 1993*, Springer-Verlag, pp 21-40, 2001.

[2] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art", *Evolutionary Computation*, 8 (2), pp 125-147, 2000.

[3] C. M. Fonseca and P. J. Fleming, "Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation", *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, 28(1):26–37, 1998.

[4] N. Srinivas and K. Deb, "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms", *Evolutionary Computation*, 2(3):221–248, 1994.

[5] T. Hanne, "Global Multiobjective Optimization with Evolutionary Algorithms: Selection Mechanisms and Mutation Control", *EMO 2001, LNCS 1993*, Springer-Verlag, pp 197-212, 2001.

[6] A. Osyczka and S. Krenich, "Evolutionary Algorithms for Multicriteria Otpimization with Selecting a Representative Subset of Pareto Optimal Solutions", *EMO 2001, LNCS 1993*, Springer-Verlag, pp 141-153, 2001.

[7] J. Valenzuela and A. E. Smith, "A Seeded Memetic Algorithm for Large Unit Commitment Problems", *Journal of Heuristics*, Kluwer Academic, 8, p173-195, 2002.

[8] R. A. Arapoglu, B. A. Norman and A. E. Smith, "Locating Input and Output Points in Facilities Design – A Comparison of Constructive, Evolutionary, and Exact Methods", IEEE Transactions on Evolutionary Computation, Vol. 5, no. 3, pp 192-203, 2001.

[9] R. R. Hill, "A Monte Carlo Study of Genetic Algorithm Initial Population Generation Methods", *proceedings of the 1999 Winter Simulation Conference,* pp 543-547.

[10] B. W. Kernighan and S. Lin, "An efficient Heuristic Procedure for Partitioning Graphs", *Bell System Technical Journal*, 1970

[11] C. Sechen, "Chip-Planning, Placement and Global Routing of Macro/Custom Cell Integrated Circuits Using Simulated Annealing", *Proceedings of the 25th Design Automation Conference*, pp 73-80, 1988.

[12] H-M. Chen, H. Zhou, F.Y. Young, D.F. Wong, H.H. Tang and N. Sherwani, "Integrated Floorplanning and Interconnect Planning", *proceedings of the International Conference on Computer Aided Design*, pp 354-357, 1999

[13] H. Esbensen and E.S. Kuh, "EXPLORER: An Interactive Floorplanner for Design Space Exploration", *proceedings of the European Design Automation Conference*, 1996.

[14] J.H. Holland, "Adaptation in Natural and Artificial Systems", *Univ. of Michigan Press*, Ann Harbor, 1975

[15] K.D. Boese, A.B. Kahng and G. Robins, "High Performance Routing Trees With Identified Critical Sinks", *proceedings of the 30th Design Automation Conference*, pp 182-187, 1993.

[16] D.F. Wong and C.L. Liu, "A New Algorithm for Floorplan Design", proceedings of the 23rd Design Automation Conference, pp 101-107, 1986.