International Journal of Computer Applications (0975 – 8887)
Volume 19– No.7, April 2011

Multi-Objective Optimization of Standard Cell Placement using Memetic Algorithm

Aaquil Bunglowala
Department of EC
Sanghvi Inst. of Mangt & Sc.,
Indore MP 453331

Dr. Brijmohan Singhi
Department of EC
Medicaps Inst. of Tech. & Mangt,
Indore MP 453331

Dr. Ajay Verma
Department of EI
IET, DAVV
Indore MP 452017

ABSTRACT

Beyond the optimization of single parameter (usually the wire-length) in Standard Cell Placement (SCP), focus in the present work is laid on the optimization of speed, power, and the wire length. As discussed in our previous work of hybrid algorithms for single objective optimization of SCP the main advantage of hybridization is the improvement in convergence speed to Pareto front although it leads to increase in computation time per generation. Memetic Algorithm (MA) is a hybrid of Genetic Algorithm (GA) & Local search (LS) wherein we need to strike a right balance of the two for optimum solution. In this paper we work on our previous GA based multi-objective SCP algorithm [2] for simultaneous optimization of power, speed and wire length while maintaining the layout width as constant by choosing initial population to be alleles of high fitness value and apply proper local search to all the members of initial population. Further we compare the results with previously established GA based algorithm by applying the two on ami20, ami33 and ami120 cell library instances. The Memetic algorithm is found to give better results with 10% improvement in wire-length, 7.5% lesser delay and power consumption reduction by nearly 6%.

Keywords: Memetic Algorithm, Pareto front, alleles, local search, fitness

1. INTRODUCTION:

Standard cell placement problem is computationally very hard. It is a NP-hard problem. These problems cannot be solved in polynomial time. Trying to evaluate every possible placement to determine the best one takes time proportional to the factorial of number of cells[7]. Therefore heuristic algorithms are used to search through a large number of candidate placement configurations. Multi-objective optimization algorithms are even more time consuming. One promising approach for improving convergence to Pareto front is the use of local search in multi-objective GA (MOGA). Hybridization of GA with local search is often referred to as Memetic Algorithm (MA). In Multi-objective GA based Local Search (MOGALS) a scalar fitness function with random weights is used for selection of parents and local search is used to generate their offspring [5]. An improved MOGALS is proposed by avoiding use of roulette wheel selection over entire population. Here parents are selected randomly from pre-specified number of best solutions with respect to scalar fitness function with current weights.

In this paper we use Algorithm-II [2] as the basic MOGA and introduce concept of MOGALS wherein local search procedure is applied to each offspring using the same scalar fitness function in the selection of their parents.

2. MA-I [SCP using MOGALS]

Here Memetic Algorithm based on Genetic Algorithm and local search is presented with as objective of optimizing power consumption, delay and wire length taking fixed width of the layout as a constraint. The problem is specified in our work [3]. The cost functions are formulated for all the three parameters separately; and subsequently an overall scalar fitness function is computed.

Wire length Cost function: Total wire length is computed by adding estimated wire length of each net in the circuit:

$$Wire_{\cos t} = \sum_{x \in N} W_{Lx} \tag{1}$$

where, W_{Lx} is the wire length estimated for net x and N is total no. of nets in circuit.

Power Cost Function: Power consumption P_x of net x in a circuit can be given by:

$$P_x = \tfrac{1}{2} C_x V_{DD}^2 . f . S_x . \beta$$

where, C_x is total capacitance of net x, V_{DD} is the supply voltage, f is the clock frequency, Sx is the switching probability of net x; and β is a technology dependent constant. For fixed supply voltage and fixed clock frequency P_x can be approximated as:

$$P_x \cong C_x . S_x$$

The value of C_x for cell x is given by:

$$C_x = C_g + C_r$$

where, C_g is gate capacitance and C_r is interconnect capacitance at the output node of the cell x. At the cell placement level, only interconnect capacitance can be manipulated and that too, is dependent on wire length of net x.

So, $P_x \cong L_x . S_x$

Therefore,

$$Power_{\cos t} = \sum_{x \in N} P_x = \sum_{x \in N} L_x . S_x \tag{2}$$

Delay Cost function: Delay cost is determined by considering the delay along the longest path in a circuit.

$$Delay_{longest-path} = \sum_{x=1}^{n-1} (D_{swtx} + D_{\text{int}\,x}) \tag{3}$$

where, D_{swtx} is the switching delay of the cell driving net x and D_{intx} is the interconnect delay of the same net.

Interconnect delay is the one is affected in the placement phase. Switching delay is technology dependent and remains unchanged.

To compute a cost function representing the effect of all the three objectives in the form of single quantity, a scalar fitness function to be minimized was used in both the selection of parents and also the local search for their offspring as:

$$fitness = (w_w * Wire_{cost}) + (w_p * Power_{cost}) + (w_d * Delay_{longest-path}) \quad (4)$$

where w_w, w_p, w_d are the finite positive weights of each cost function randomly selected for each pair of parents such that

$$w_w + w_p + w_d = 1$$

GA based approach again is divided into five stages of operation:

1. Initial Stage
2. Fitness Evaluation
3. Parent Preference and Crossover
4. Selection & Mutation
5. Local Search

2.1. INITIAL STAGE:

2.1.1 Chromosome Encoding:

In order to process a solution by GA, it is necessary to represent it in the form of a chromosome. A placement solution is the arrangement of cells in a 2-D layout. So the solution is represented in the form of a 2-D grid.

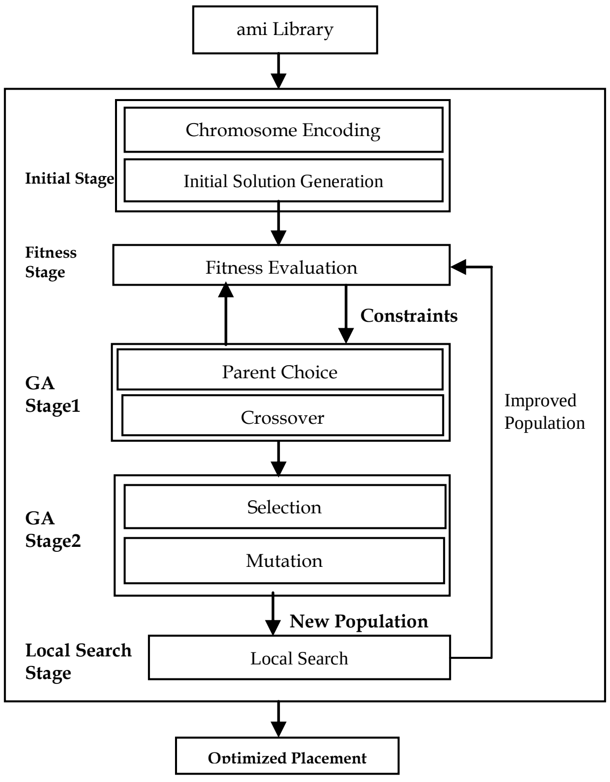

Figure 1: Flow of MOGALS based Algorithm

Due to varying widths of the cells in a circuit, the rows may not have equal number of cells. Consider a circuit comprising of 12 cells: 1, 2 , 3 , . . , 12. A probable layout may be:

7	3	4	11
8	12	6	
5	2	1	
9	10		

The above layout is made after computing the average row width. This average row width is divided by the smallest cell width to compute the maximum number of possible locations in a row. Assume 4 locations and let it be known from the min-cut placer that there are 4 rows in the layout.

2.1.2 Initial Solution Generation:

Looking at the probable placement of 12 cells as above, initial solution is generated by random selection of a cell out of the 12, and placing it in the first row. Before placing a cell, it is checked whether the addition of it is violation of the width constraint. If any violation, place it at the start of next row. All the cells are thus placed on the layout. As a result, there are four empty locations as shown above. To make it a packed grid, we fill the empty locations by dummy cells represented by negative integers as shown below.

2.2. FITNESS EVALUATION

The fitness of a solution is a measure of its proximity to the optimal solution. Higher the fitness value of a solution, closer it is to the optimal solution. In our implementation, an initial solution is assigned a fitness value of 0 whereas the optimal solution is assigned a fitness value of 1. The purpose is to normalize the fitness value of any solution in range [0,1].

7	3	4	11
8	12	6	-1
5	2	1	-2
9	10	-3	-4

2.3. PARENT PREFERENCE & CROSSOVER

2.3.1 Parent Preference:

In this algorithm roulette wheel scheme [6] of parent selection is used. An individual chromosome is selected with a probability proportional to its fitness value. This scheme allows the individuals having low fitness values to be selected with a low probability.

2.3.2 Crossover:

Each allele in the chromosome representation is distinct and this property must be retained from generation to generation for a chromosome to represent a valid solution. Therefore, simple crossover techniques are not to be used as it may generate duplicates. Therefore we use other types of existing crossover operators like order crossover, PMX etc and its modified form. Restricted PMX (RPMX). In RPMX, PMX is applied between first and second parent and then between second and first parent. As a result, two off-springs are generated and the better offspring with a high probability is chosen.

The crossover operation is performed with a high probability p_h. Different high crossover probabilities are attempted to choose parents. After choosing two parents, a random number n_r in the range [0,1] is generated, and if $n_r < p_h$

crossover is applied. If it fails, another set of parents is chosen and the process is repeated. This process ensures that same number of offsprings is generated in each iteration and total number of off-springs generated are equal to the population size. On generation of a new offspring if the width cost of the circuit is violating the width constraint, it is discarded. This process continues until the number of offspring is equal to the initial population.

2.4. SELECTION & MUTATION

2.4.1 Selection:

Here the selection is performed before the mutation. This encourages the diversity in the population by ensuring the transfer of mutation effect into next generation. We experimented this with different selection schemes including roulette, elitist roulette, elitist-random, and extended-elitist-random schemes.

a. In elitist roulette-random selection, the best half of the chromosomes are selected and the remaining half are selected using roulette wheel.

b. In elitist roulette selection, the best chromosome is selected among parents and off springs, and the remaining are selected using roulette wheel.

c. In extended elitist-random selection, the best half of the chromosomes are selected and the remaining half are selected randomly.
d. Similarly, in elitist-random selection, the best one is selected from parents and off-springs, while the remaining are randomly selected.

2.4.2 Mutation:

Here, mutation with a dynamic probability P_D is used. It is a function of the diversity of the population selected for the next generation i.e. (k+1)th generation. The standard deviation of the population in (k+l)th generation used as a measure of the diversity. It is to increase the mutation probability when population tends to lose diversity.

We implemented mutation as a series of random pair-wise interchanges. The number of inter-changes was taken depending on the size of the circuit. A random fraction *fract* between 0.02 and 0.04 was generated and *fract x n* interchanges were made, where *n* was the total number of cells in the circuit.

2.5. LOCAL SEARCH:

Here, a local search procedure is applied to each solution in the current population using the scalar fitness function given in equation (4). The weight vector used for parent selection is reused for local search excluding the initial solution where random weight vector is used. Local search terminates when there is no improvement in the solution space of k solution randomly selected from the neighborhood of the current solution. After local search is applied to all the solutions of the population the current solution is replaced with improved solutions. The algorithm is a Lamarckian Multi-objective Memetic Algorithm.

MOGALS terminates when a pre-specified number of solutions are examined. In local search part of the algorithm, a neighbor is randomly generated from neighborhood of current solution. This replaces the current solution if it is better using first improvement strategy. The algorithm continues in the similar fashion once the current solution is updated. In this algorithm all non dominates solutions are stored in the secondary population with no restriction on its size.

3. RESULTS & DISCUSSION

The algorithms were implemented in C language. Nine different test-cases were considered from ami standard cell library. These test cases were mapped over 0.18 micron, 5 metal layer standard cell library [7].
In Algorithm-II [2] layout width was constrained not to exceed 1.25 times the average row width. This constraint was satisfied in obtaining the results shown here. In case of MOGALS, results were computed over 20,000 generations. The population size was kept 40 chromosomes and RPMX crossover with a probability equal to 0.95 was applied. Extended-elitist random selection was used to attain best final results. Table-1 summarizes the results of GA & MOGALS based algorithms. The circuits are listed in the increasing order of the number of contained cells K. Here $Wire_{Cost}$, $Power_{Cost}$ and $Delay_{Cost}$ represent the wire length, power and delay costs respectively.

4. CONCLUSION

As concluded in our previous work [2] Algorithm-I produced average solutions with 20% better timing, 12% less power consumption and is 1.8 times faster than the timing driven AMOEBA-SOCE. We subsequently presented an iterative approach in Algorithm-II based on GA for multi-objective optimization of VLSI standard cell placement. The fuzzy logic is used to integrate the three objectives wire length, delay and power into a scalar cost function. Algorithm-II showed marginal improvement in all the parameters over Algorithm-I for all the three levels of complexity of ami standard cell libraries. With this as the basis we attempted integration of GA with local search in MA and observed that for the same test case of 0.8 micron ami based standard cell library the observed results were encouraging with marginal improvement in all the three parameters as compared to Algorithm-II including 10% improvement in wire length 7.5% lesser delays and power consumption reduction by nearly 6%. This is at a cost of increased computational time and can be further reduced by restrictive application of LS search and will be attempted later.

5. REFERENCES

[1] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements" Proc.of Design Automation Conference,pp.477-482, 2000.

[2] Bunglowala, A., Singhi, B. M., “Standard Cell Placement using Iterative & Constructive Heuristics for Multi-Objective Optimization”, published in International Journal of Electronics Engineering, 2(1),2010,pp. 131-135.

[3] Bunglowala, A., Singhi, B. M., “Performance Evaluation and Comparison and Improvement of Standard Cell Placement Techniques in VLSI Design”, ICETET-IEEE Computer Society, pp. 468-473, 2008.

[4] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Application in

VLSI Design," In Proc. of Design Automation Conference, pp. 526-529, 1997.

[5] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," in Proc. of 3rd IEEE Conf. on Evolutionary Computation, Nagoya, Japan, pp 119-124, 1996

[6] J. Y Sayah et. al., "Design planning for high-performance ASICs", In IBM Journal of Research and Development, Vol. 40, No. 4, pp. 431-452, 1996.

[7] K. Shahookar and P. Mazumder. VLSI Cell Placement Techniques. *ACM Computing Surveys*,2(23):143-220, June 1991.

[8] Sadiq M. Sait and Habib Youssef. *Iterative ComputerAlgorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* IEEE Computer Society Press, California, December 1999.

[9] Virtual-Silicon Technology Inc., http://www.virtual-silicon.com.

[10] X. Yang, B. Choi, and M. Sarrafzadeh, "Timing-Driven Placement using Design Hierarchy Guided Constraint Generation," In Proc. Int'l Conference on Computer-Aided Design, pp. 177-180, 2002

Table 1: Comparison of MOGALS and GA based Algorithm-II results

Std. Cell instance	Complexity	Wire length WL (μm)		Delay D (nsec)		Power P (mW)	
		Algorithm-II	MOGALS	Algorithm-II	MOGALS	Algorithm-II	MOGALS
ami20	Minimum	456422	411020	5.246	4.962	51.17	48.65
ami20	Moderate	537840	484044	6.489	6.124	57.64	54.76
ami20	Maximum	654625	589162	9.773	9.088	77.26	73.36
ami33	Minimum	578461	520622	7.281	6.762	64.98	61.72
ami33	Moderate	717545	645796	10.959	10.151	78.14	74.16
ami33	Maximum	869442	782488	15.232	14.114	91.02	86.38
ami120	Minimum	1193433	1074110	26.027	24.143	114.52	107.46
ami120	Moderate	1261246	1135220	29.129	26.824	120.65	112.54
ami120	Maximum	1495627	1346104	34.438	31.537	135.86	126.82

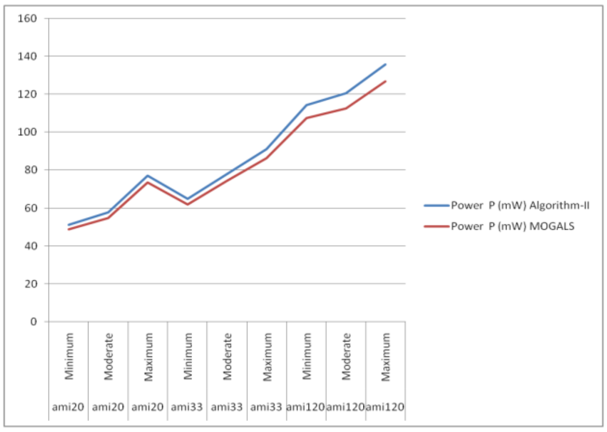

Figure 2: Power Comparison MOGALS v/s Algorithm-II

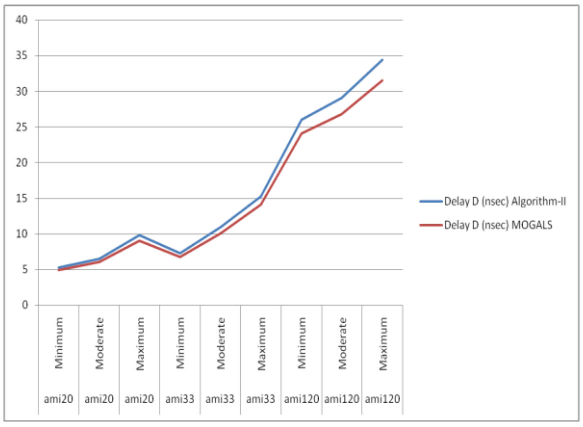

Figure 3: Delay Comparison MOGALS v/s Algorithm-II

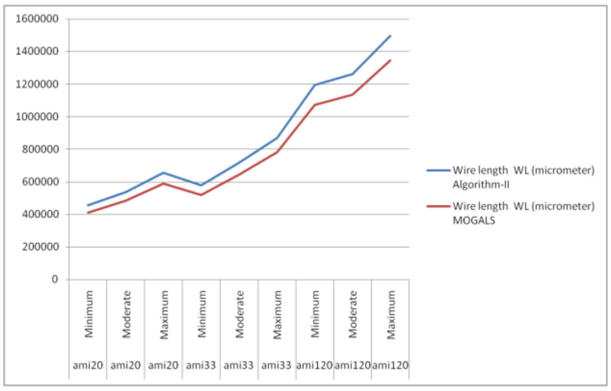

Figure 4: Wire Length Comparison MOGALS v/s Algorithm-II