Journal of Computer Science 9 (8): 943-948, 2013
ISSN: 1549-3636

doi:10.3844/jcssp.2013.943.948 Published Online 9 (8) 2013 (http://www.thescipub.com/jcs.toc)

AN EFFICIENT CELL PLACEMENT USING GRAVITATIONAL SEARCH ALGORITHMS

[1]Rose Al Qasem and [2]Taisir Eldos

[1]Department of Computer Engineering, Al Balqa Applied University, Amman, Jordan

[2]Department of Computer Engineering, Jordan University of Science and Technology, Ar Ramtha, Irbid Jordan

Received 2013-06-01, Revised 2013-06-24; Accepted 2013-06-25

ABSTRACT

In modern chip design, cell placement is a stage in which cells representing well-defined functions are assigned physical locations, in a way that optimizes the total area and routing length. Cell placement is an NP-complete problem and the exact solution is generally far from reach for a practically sized instance. Hence, diversified heuristic algorithms are used to solve this problem. In this study, we adapted a recently introduced evolutionary search algorithm called Gravitational Search Algorithm (GSA) to this problem. Experiments show that the proposed algorithm delivers good performance; good solution quality and likelihood of optimality within reasonably small amount of time.

Keywords: Cell Placement, VLSI Circuit, Optimization Algorithms, Gravitational Search

1. INTRODUCTION

Cell placement is one of four consecutive steps in physical design process of VLSI circuits, namely: Partitioning, placement, routing and compaction. In the placement stage, the description of the physical layout of the chip is introduced, by assigning geometric coordinates to the cells. The objective of the placement algorithm is to find a layout that minimizes a cost function, whose major part is the area, but quite often involves the aspect ratio, to make the chip as close to square as possible and hence increase the die yield.

Approaches to solve cell placement problem are generally classified into two classes; constructive and iterative improvement methods. Several heuristic optimization strategies for solving placement problem have been implemented via a set of diversified algorithms; evolution-based placement like Genetic Algorithms used by Valenzuela and Wang (2002) and Simulated Annealing used by Khorgade *et al*. (2009) and a comprehensive summary of those strategies is presented by Shahookar and Mazumder (1991).

Gravitational Search Algorithms (GSAs) are novel heuristic optimization algorithms based on the gravitational law and laws of motion, introduced by Rashedi *et al*. (2009a) and researched in the past few years, as a flexible and well-balanced strategy to improve exploration and exploitation methods. Rashedi *et al*. (2009b) developed binary gravitational search algorithm to solve different nonlinear problem. A new multi-objective gravitational search algorithm was proposed by Hassanzadeh and Rouhani (2010). The GSA shows satisfactory results for solving many problems in a various applications; an improved Gravitational Search Algorithm was used by Li and Zhou (2011) for solving the parameters identification of hydraulic turbine. Zhao (2011) applied Gravitational Search Algorithm to improve the effect of image enhancement. Xiao and Cheng (2011) implemented an approach based on Gravitational Search Algorithm to solve the DNA sequence design problem.

In this study, we formalize a solution to the cell placement problem using the gravitational search methodology, with the intention compare its performance with well know evolutionary algorithms in future work. The results show that the algorithm can improve the solution quality in a reasonable amount of time. This study is organized as follows: Section 2.1

Corresponding Author: Rose Al Qasem, Department of Computer Engineering, Al Balqa Applied University, Amman, Jordan

gives a formal description of the GSA theory, Section 2.2 gives a brief description of the cell placement problem, section 2.3 demonstrates the proposed gravitational search algorithm for cell placement, section3 presents and discuses the results of the experiments and section 4 wraps up our work.

2. MATERIALS AND METHODS

2.1. Gravitational Search Algorithm (GSA)

The Gravitational Search Algorithm (GSA) was proposed by Rashedi *et al*. (2009a), as a simulation of Newton's gravitational force behaviors. In this algorithm, possible solutions of the problem in hand are considered as objects whose performance (quality) is determined by their masses, all these objects attract each other by the gravity force that causes a global movement of the objects towards the objects with heavier masses. The position of each object corresponds to a solution of the problem and inertial masses are determined by a fitness function. The heavy masses, which represented a good solutions, move more slowly than lighter ones, this represents the exploitation of the algorithm.

2.2. The GSA Behavior

The GSA starts with a set of agents, selected at random or based on some criteria, with certain positions and masses representing possible solutions to a problem and iterates by changing the positions based on some values like fitness function, velocity and acceleration that gets updated in every iteration. To relate those values and parameters, let us demonstrate the relations among them.

In a system with N agents, the position of the i^{th} agent is defined as Equation 1:

$$X_i = \left(x_i^1,...,x_i^d,...,x_i^n\right) \text{ for } i = 1,2,...,N \quad (1)$$

where, x_i^d present the position of the i^{th} agent in the d^{th} dimension and n is dimension of the search space.

At the time t a force acts on mass i from mass j. This force is defined as follows Equation 2:

$$F_{ij}^d = G(t)\frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}+E}\left(x_i^d(t)x_i^d(t)\right) \quad (2)$$

Where:

M_{aj} = The active gravitational mass of agent j,
M_{pi} = The passive gravitational mass of agent i,
G(t) = Gravitational constant at time t, ε is a small constant and
$R_{ij}(t)$ = The Euclidian distance between two agents i and j Equation 3:

$$R_{ij}(t) = \| X_i(t), X_j(t) \| \quad (3)$$

The total force acting on $mass_i$ in the d^{th} dimension in time t is given as follows Equation 4:

$$F_i^d(t) = \sum_{jeKbest,\, j\neq i}^{N} rand_j F_{ij}^d(t) \quad (4)$$

where, $rand_j$ is a random number in the interval [0, 1], K best is the set of first K agents with the best fitness value.

The acceleration related to mass i in time t in the d^{th} dimension is given as follows Equation 5:

$$a_i^d = \frac{F_i^d(t)}{M_{ii}(t)} \quad (5)$$

where, M_{ii} is the inertial mass of i^{th} agent.

The next velocity of an agent could be calculated as a fraction of its current velocity added to its acceleration. Position and velocity of agent is calculated as follows Equation 6 and 7:

$$v_i^d(t+1) = rand_i^d v_i^d(t) + a_i^d(t) \quad (6)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (7)$$

where, $rand_i$ is a uniform random variable in the interval [0, 1].

Gravitational constant, G, is initialized at the beginning of the search and will be reduced with time to control the search accuracy as follows Equation 8:

$$G(t) = G_o e^{-\alpha\frac{t}{\tau}} \quad (8)$$

where, T is the number of iteration, G_0 and α are given constant.

The gravitational mass and the inertial mass are updated by the following Equation 9-11:

$$M_{ai} = M_{pi} = M_{ii} = M_i, \quad i = 1,2,....,N \quad (9)$$

$$m_i(t) = \frac{fit_i(t) - warst(t)}{best(t) - worst(t)} \quad (10)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{i=1}^{N} m_j(t)} \quad (11)$$

where, $fit_i(t)$ represent the fitness value of the agent i at time t and, worst(t) and best(t) are given as follows for a minimization problem Equation 12 and 13:

$$best(t) = \min_{j \in (1,...N)} fit_j(t) \quad (12)$$

$$worst(t) = \max_{j \in (1,...N)} fit_j(t) \quad (13)$$

2.3. Cell Placement Problem

We use the Normalized Polish Notation (RPN) which presented by Wong and Liu (1986) to describe any arrangement representing a possible solution; for n cells, a string with n modules (cells) and n-1 operators of the * or + type, to mean above or next to. As an example, the string (2 3 * 1 + 4 5 + 6 7* + *) is an encoding for the arrangement in **Fig. 1**. Here, relaxed means the case where the area is that of the minimal rectangle enclosing the cells, while the restricted means the case where the area is that of the minimal square enclosing the cells.

Such a configuration is an agent in gravitational search algorithm; new agents are generated from the existing ones by applying certain operators which are described by Wong and Liu (1986) and Cohoon *et al*. (1991). New solutions are assigned fitness values that reflect their quality. We proposed the following fitness measure Equation 14:

$$Fitness = \alpha\frac{A}{SL} + (a-\alpha)\frac{S}{L} \quad (14)$$

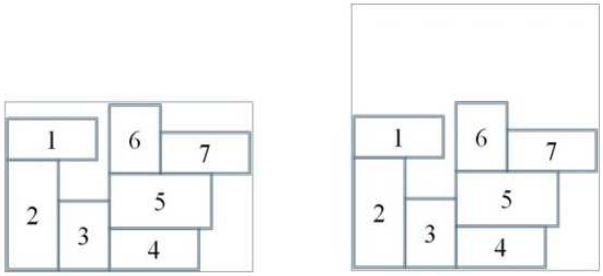

Fig. 1. (a) Relaxed and (b) Restricted area

where, L and S are the long and short sides of the rectangle enclosing all the cells and A is the algebraic sum of the areas of all cells regardless of the placement and the product SL represents the area associated with the solution. The factor α is a number between 0 and 1, introduced to dictate the relative significance of the aspect ratio to the actual area; to favor square arrangements we used smaller values of α. If α = 1 then aspect ratio is not optimized.

2.4. Gravitational Search Algorithm Based Cell Placement

Cell placement can be viewed as a two-dimensional bin packing problem, where the goal is to arrange the cells in a way that reduces the area and producing near square die while providing enough space for efficient routing. In this sense, we propose a new algorithm for cell placement problem by means of GSA, in which each mass will be an agent looking for an optimal solution in the search space.

Since cell placement needs meet simultaneously several constraints, it is difficult to be solved by the traditional GSA. For this reason, the definition of distance between solutions (positions) and their update are modified as will be shown in the following procedure.

The equations which used in the procedure are given as follow Equation 15-21:

$$mass_i(t)\frac{fitness_i(t) - worst}{\sum_{j=1}^{N}\left(fitness_j(t) - worst\right)} \quad (15)$$

$$force_{ij}(t) = G(t)\frac{mass_i(t) \times mass_j(t)}{distance_{ij}(t) + \in} \quad (16)$$

Where:

$$\text{€} = 0.1$$

$$G(t) = G_{ini}\left(1 - \frac{iteration}{total\ iteration}\right) \quad (17)$$

Where:

$$G_{ini} = 100$$

$$total\ force_i(t) = \sum_{j=1}^{N} rand_j \times force_{ij}(t) \quad (18)$$

$$\text{acceleration}_i(t) = \frac{\text{total force}_i(t)}{\text{mass}_i(t)} \quad (19)$$

$$\text{velocity}_i(t+1) = \text{rand}_i \times \text{velocity}_i(t) + \text{acceleration}_i(t) \quad (20)$$

$$\text{propapility}_i = |\tanh(\text{velocity}_i(t+1)|) \quad (21)$$

However, the position updating Equation (7) cannot be applied in our case, because we are working in a string form to present the solution. Therefore, Rashedi *et al.* (2009b) proposed the Binary GSA for nonlinear problem, which has the same formulation presented above, but with a different equation for updating the position of each agent. In order to update our solution, the formulation of binary GSA is modified as shown in step 3.6 of the following algorithm.

However, stopping criteria can be based time budget, completing fixed number of iterations, or reaching a state after which the expected future improvement does not justify the time to be invested.

The Algorithm Outline

The gravitational search algorithm is outlined as follows:

1. Generate initial population of N agents at random
2. Compute G(t), Best Fitness and Worst Fitness
3. For each agent i, do:
   - 3.1. Evaluate Fitness_i
   - 3.2. Evaluate Mass_i
   - 3.3. Evaluate Force of Mass_i
   - 3.4. Evaluate Acceleration of Mass_i
   - 3.5. Update Velocity of Mass_i
   - 3.6. Find new Position of Agent_i

     If (Probability_i > Small Threshold)
     {
     If (Rand_i < Probability_i)
     Then Pair Solution_i with the Best Fit Solutions
     Else impose some minor change to Solution_i
     }
4. If Stopping Criteria Not Met, Go To 2 Else Stop

3. RESULTS AND DISCUSSION

Briefly, the quality measures are: Number of successes in a set of runs (success rate), average fitness value for a set of runs and number of iterations employed (elapsed time).

In order to measure the quality of the proposed Algorithm for solving the cell placement problem, two studies are performed. In first study, instances from MCNC benchmark are selected. In this study, the algorithm has achieved good results for the fitness value. In the second study, the algorithm is applied on three cases of cell placement problem which independently generated and have known optimal solutions. This study shows that the algorithm has achieved high success rate for finding optimal solution.

In the first study, three benchmarks; Xerox (10 cells, 19.35 square millimeter), Ami33 (33 cells, 1.16 square millimeters) and Ami49 (49 cells, 35.44 square millimeters), were selected from MCNC and tested. The measure used is the quality in terms of wasted area and aspect ratio. The number of trials for each problem is 10 replicates, each with different random starts. The GSA is halted when 5000 iterations have passed since the last improvement was recorded.

Table 1 shows the results for the three instances of MCNC benchmark, each run 10 times with different initial population. The results indicate that our GSA achieves promising area utilization in the best, worst and mean wasted area. **Figure 2** shows the search progress of the algorithm; Waste area versus Iterations number (Ami49 upper, Ami33 lower). It is clear that the algorithm can achieve better value through the passage of time.

As mentioned, the second study has three optimal cases for 10, 20 and 30 cells as shown in **Fig. 3**. The algorithm runs on each case 6 times. The algorithm is brought to stop if the optimal solution is achieved or the number of iterations equals 15000, 30000 and 45000 for case1, case2 and case3 respectively. The used measures are the likelihood of optimality or success rate. **Table 2** shows the result for the success rate for each case.

Table 1. Benchmark, mean, best and worst waste area percentage (10 runs each)

	#Cells	Iterations	Mean	Best/aspect ratio	Worst/aspect ratio
Xerox	10	15,000	0.054	0.042/1.63	0.067/1.05
Ami33	33	35,000	0.072	0.061/1.45	0.091/1.67
Ami49	49	50,000	0.098	0.081/1.56	0.13/1.21

Table 2. Success rate of artificial problems

Number of cells	Number of iterations	Success rate
10	15,000	6 out of 6
20	30,000	3 out of 6
30	45,000	2 out of 6

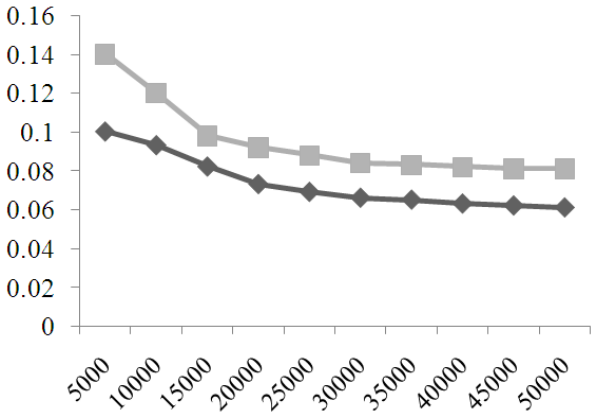

Fig. 2. Search progress; waste area versus iterations (Ami49 upper, Ami33 lower)

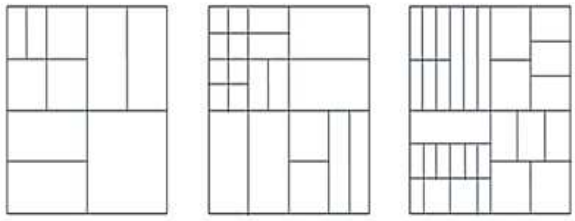

Fig. 3. Artificial Problems for Known Optimal Solutions; case1, case2 and case3

In the first case, an instance is created by cutting a square into 10 variable size blocks with total area of 16 square units. The success rate for our GSA is 100% over 6 different random runs; achieved the optimal solution in every run. In the second case, an instance is created by cutting a square into 20 variable size blocks with total area of 64 square units. The success rate for our GSA based cell placement is 50%, in the third case; an instance is created by cutting a square into 30 variable size blocks with total area of 144 square units. The success rate dropped to 33%. The algorithm is robust for problems with small size and need to be enhanced to get better results for medium and large size problems.

4. CONCLUSION

This study proposed a new algorithm which based on Gravitational Search Algorithm to solve the Cell Placement problem. The efficiency of this algorithm appears in terms of success rate, likelihood of optimality, average of fitness value for a set of runs and the minimum of wasted area. Gravitational Search Algorithm seems to be an interesting algorithm for cell placement, since there is no consensus on which method is better than the other to solve a given problem, we will compare the proposed algorithm with other optimization algorithms in future work.

5. REFERENCES

Cohoon, J.P., S.U. Hedge, W.N. Martin and D.S. Richards, 1991. Distributed genetic algorithms for the floorplan design problem. IEEE Trans. Comput. Aided Design, 10: 483-492. DOI: 10.1109/43.75631

Hassanzadeh, H.R. and M. Rouhani, 2010. A multi-objective gravitational search algorithm. Proceedings of the 2nd International Conference on Computational Intelligence, Communication Systems and Networks, Jul. 28-30, IEEE Xplore Press, Liverpool, pp: 7-12. DOI: 10.1109/CICSyN.2010.32

Khorgade, M., A.Y. Deshmukh, P. Bajaj and A.G. Keskar, 2009. Optimization of cost function with cell library placement of VLSI circuits using simulated annealing. Proceedings of the 2nd International Conference on Digital Object, Dec. 16-18, IEEE Xplore Press, Nagpur, pp: 173-178. DOI: 10.1109/ICETET.2009.163

Li, C. and Z. Zhou, 2011. Parameters identification of hybdraulic turbine governing system using improved gravitational search algorithm. Energy Conversion Manage., 52: 374-381. DOI: 10.1016/j.enconman.2010.07.012

Rashedi, E., H. Nezamabadi-Pour and S. Saryazdi, 2009a. BGSA: Binary gravitational search algorithm. Nat. Comput. 9: 727-745. DOI: 10.1007/s11047-009-9175-3

Rashedi, E., H. Nezamabadi-Pour and S. Saryazdi, 2009b. GSA: A gravitational search algorithm. Inform. Sci., 179: 2232-2248. DOI: 10.1016/j.ins.2009.03.004

Shahookar, K. and P. Mazumder, 1991. VLSI cell placement techniques. ACM Comput. Survey, 23: 143-220. DOI: 10.1145/103724.103725

Valenzuela, C.L. and P.Y. Wang, 2002. VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions. IEEE Trans. Evolut. Comput., 6: 390-401. DOI: 10.1109/TEVC.2002.802872

Wong, D.F. and C.L. Liu, 1986. A new algorithm for floorplan design. Proceedings of the 23rd Conference on Design Automation, Jun. 29-2, IEEE Xplore Press, pp: 101-107. DOI: 10.1109/DAC.1986.1586075

Xiao, J. and Z. Cheng, 2011. DNA sequences optimization based on gravitational search algorithm for reliable DNA computing. Proceedins of the 6th International Conference on Bio-Inspired Computing: Theories and Applications, Sept. 27-29, IEEE Xplore Press, Penang, pp: 103-107. DOI: 10.1109/BIC-TA.2011.12

Zhao, W., 2011. Adaptive image enhancement based on gravitational search algorithm. Proc. Eng., 15: 3288-3292. DOI: 10.1016/j.proeng.2011.08.617