On The Performance of the Gravitational Search Algorithm

Taisir Eldos
Department of Computer Engineering
Faculty of Computer and Information Technology
Jordan University of Science and Technology
Irbid - Jordan

Rose Al Qasim
Department of computer Engineering
Faculty of Engineering and Technology
Al Balqa Applied University
Amman, Jordan

***Abstract*—Gravitational Search Algorithms (GSA) are heuristic optimization evolutionary algorithms based on Newton's law of universal gravitation and mass interactions. GSAs are among the most recently introduced techniques that are not yet heavily explored. An early work of the authors has successfully adapted this technique to the cell placement problem, and shown its efficiency in producing high quality solutions in reasonable time. We extend this work by fine tuning the algorithm parameters and transition functions towards better balance between exploration and exploitation. To assess its performance and robustness, we compare it with that of Genetic Algorithms (GA), using the standard cell placement problem as benchmark to evaluate the solution quality, and a set of artificial instances to evaluate the capability and possibility of finding an optimal solution. Experimental results show that the proposed approach is competitive in terms of success rate or likelihood of optimality and solution quality. And despite that it is computationally more expensive due to its hefty mathematical evaluations, it is more fruitful on the long run.**

Keywords—Optimization; Gravitational Search; Genetic Algorithms; Cell Placement

I. INTRODUCTION

GSA is a heuristic stochastic swarm-based search algorithm in the field of numerical optimization, based on the gravitational law and laws of motion. Like many other nature inspired algorithms, it needs refinements to maximize its performance in solving various types of problems. In addition to the problem encoding that sometimes can be a challenge, fine tuning its parameters play a significant role balancing the search time versus solution quality. This algorithm is relatively recent and not heavily explored.

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.

II. LITERATURE REVIEW

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 [5] and Simulated Annealing [6], and a comprehensive summary of those strategies is presented in [1].

Gravitational Search Algorithms (GSAs) are novel heuristic optimization algorithms introduced in [2], and researched in the past few years, as a flexible and well-balanced strategy to improve exploration and exploitation methods. In [3], the binary gravitational search algorithm was developed to solve different nonlinear problem. A new multi-objective gravitational search algorithm was proposed in [4]. The GSA shows satisfactory results for solving many problems in a various applications; Solving Symmetric Traveling Salesman Problem [7], solving the flow shop scheduling problem [8], in feature selection [9], image enhancement [10], solving DNA sequence design problem [11], and optimize the filter modeling parameters [12]. A hybrid algorithm was derived from both Genetic Algorithms and Gravitational Search Algorithm for feature set selection [13].

In this paper, we enhance our implementation of the gravitational search technique, to solve the cell placement, 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 paper is organized as follows: Section 2 gives a formal description of the GSA theory, Section 3 gives a brief description of the cell placement problem, section 4 demonstrates the proposed gravitational search algorithm for cell placement, In section 5 we discuss the performance of this algorithms in solving standard problems as compared to the well know genetic algorithm, and section 6 wraps up our work.

III. METHODOLOGY

The Gravitational Search Algorithm (GSA) was proposed by Rashedi [2], 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.

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:

$$X_i = \left(x_i^1, \dots, x_i^d, \dots, x_i^n\right) \; for \;\; i = 1,2, \dots, N \tag{1}$$

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:

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

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

$$R_{ij}(t) = \left\| X_i(t). X_j(t) \right\| \tag{3}$$

The total force acting on $mass_i$ in the d[th] dimension in time t is given as follows:

$$F_i^d(t) = \sum_{j \epsilon K best, j \neq i}^{N} rand_j F_{ij}^d(t) \tag{4}$$

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:

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

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:

$$v_i^d(t+1) = rand_i \; v_i^d(t) + a_i^d(t) \tag{6}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{7}$$

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:

$$G(t) = G_0 e^{-\alpha \frac{t}{T}} \tag{8}$$

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 equations:

$$M_{ai} = M_{pi} = M_{ii} = M_i, \qquad i = 1, 2, \dots., N \tag{9}$$

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

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

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:

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

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

IV. Cell Placement Problem

We use the Normalized Polish Notation (RPN) [14] 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 Figure 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.

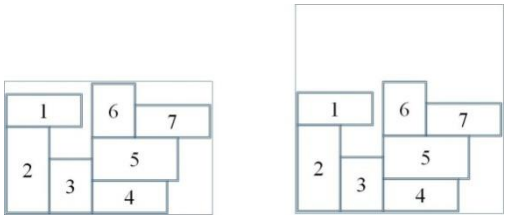

Fig. 1. (a) Relaxed and (b) Restricted area

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 in [14] and [15]. New solutions are assigned fitness values that reflect their quality. We propose the following fitness measure:

$$\mathrm{F} = \alpha\frac{\mathrm{A}}{\mathrm{SL}} + (1-\alpha)\frac{\mathrm{S}}{\mathrm{L}} \tag{14}$$

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 use smaller values of α. If α=1 then aspect ratio is not optimized.

V. Gravitational Search Algorithm Adaptation

Cell placement can be viewed as a two-dimensional bin packing problem, where the goal is to arrange a number of cells with different sizes in a way that reduces the area enclosing them 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:

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

$$force_{ij}(t) = G(t)\frac{mass_i(t) \times mass_j(t)}{distance_{ij}(t)+\epsilon}, \quad \text{where } \epsilon=0.1 \quad (16)$$

$$G(t) = G_{ini}\left(1 - \frac{iteration}{total\ iteration}\right), \quad \text{where } G_{ini}=100 \quad (17)$$

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

$$acceleration_i(t) = \frac{totalforce_i(t)}{mass_i(t)} \quad (19)$$

$$velocity_i(t+1) = rand_i \times velocity_i(t) + acceleration_i(t) \quad (20)$$

$$propapility_i = |\tanh(velocity_i(t+1)| \quad (21)$$

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 [3] 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 used as shown in step 3.6. However, the stopping criteria can be based time budget or number of iterations, or reaching a target fitness or cost function (area in cell placement), or an improvement rate less than a threshold.

VI. 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 ($Probability_i$ > 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

VII. Results

We carried two kind of tests; one on standard benchmark problems to evaluate the quality of the solutions, and another on artificial problems with known optimal solutions to measure the possibility of finding the optimal solution. The algorithm has achieved good results regarding the solution quality and success rate in finding optimal solution.

In the first study, three MCNC benchmarks; Xerox with 10 cells, Ami33 with 33 cells, and Ami49 with 49 cells, selected from MCNC and tested. Table1 1, 2, and 3 summarize the results of running the two algorithms on one of the benchmark problems in Table 1. For each case, 10 runs with different initial solutions are performed, for fixe number of iterations each. The number of iterations is set to a value proportional to the problem size. Clearly, GSA outperformed GA in the best, worst and mean waste as a measure all the time, and the aspect ratio most of the time.

TABLE I. Performance Comparison: (Xerox 10), 15000 Iterations

	GA	GSA
Best wasted area, Aspect	5.9 %, 1.83	4.2 %, 1.63
Worst wasted area, Aspect	8.3 %, 1.22	6.7 %, 1.05
Mean wasted area	7.0 %	5.4 %

TABLE II. Performance comparison: (ami33), 30000 iterations

	GA	GSA
Best wasted area, Aspect	8.6 %, 2.12	6.1 %, 1.45
Worst wasted area, Aspect	14.2 %, 1.78	9.1 %, 2.11
Mean wasted area	11.2 %	7.2 %

TABLE III. Performance comparison: (ami49), 50000 iterations

	GA	GSA
Best wasted area, Aspect	13.1 %, 2.4	8.1 %, 1.56
Worst wasted area, Aspect	18.2 %, 2.3	13.2 %, 1.21
Mean wasted area	16.1 %	9.8 %

Fig. 2 shows the progress of the two algorithms over time; wasted area of the best solution in hand after multiple thousands of iterations. Same initial set of solutions (with 42% waste best case) evolved relatively at the same rate in the first few thousands of iterations, and then the GSA starts having better progress.

The second test is carried out on three artificial problems with 10, 20 and 30 cells of known optimal solutions, where a square is split into smaller squares and rectangles to generate instances with the target size, as shown in Figure3. Both GSA and GA are run 6 times with different initial solutions.

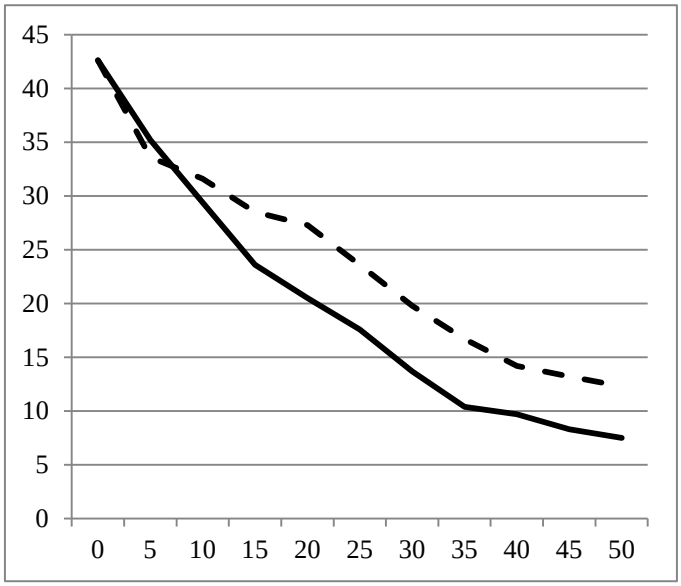

Fig. 2. Search Progress; Waste area for Ami49 versus Iterations (GSA solid, GA dashed)

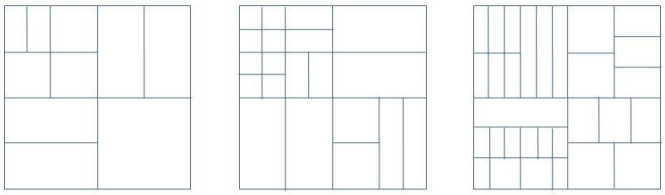

Fig. 3. Artificial Instances for Know Optimal Solutions

The algorithm is brought to stop if an optimal solution is achieved or the number of iterations equals 15000, 30000 and 50000 for the 10, 20 and 30 cells respectively. Table 4 shows the success rate or the likelihood of optimality. Again, GSA beats GA in for the small medium and large size, with significant outperformance of 100% in the 10 cells instance

TABLE IV. SUCCESS RATE OF ARTIFICIAL PROBLEMS (GSA VS. GA)

		Success Rate
No. of Cells	No. of Iterations	GSA	GA
10	15,000	6 out of 6	4 out of 6
20	30,000	3 out of 6	1 out of 6
30	45,000	2 out of 6	1 out of 6

A major drawback of thi technique is its computational requirement; each iteration needs to many computations compared to other evolutionary algorithms like genetic algorithms for example. However, the effectiveness of this search and its balance between exploration and exploitation overcome this drawback. Table 5 shows the time taken by the GSA and GA to solve a 20-cell artificial instance with known optimal solution, running with same initial set of solutions on a personal computer with moderate specs. Both algorithms are made to stop when they reach a solution with some target quality; 5%, 10% 15% and 20% of wasted area relative to the optimal area.

TABLE V. TIME REQUIREMENTS FOR 20 CELLS INSTANCE

	Time (Minutes)	
Wasted Area	GSA	GA
5%	42.3	54.3
10%	34.8	42.9
15%	31.2	30.6
20%	23.1	23.8

VIII. CONCLUSION

The GSA power of solving a relatively complex problem, such as Cell Placement, is investigated using both benchmark and artificial instances with various sizes. Comparative tests have shown that GSA outperforms GA as a well known evolutionary algorithm, in terms of solution quality, i.e. the wasted area of the best configuration, aspect ratio, and the likelihood of finding optimal solutions. It is quite significant to note that although iterations take longer time in GSA compared to GA, the total time required to achieve a target solution quality is less when we target higher quality solutions. While the two algorithms take nearly the same amount of time to find decent solutions, targeting high quality solutions; 5% waste or less, can be achieved in 75% of the time with GSA. After the first few thousands of iterations, GSA outperforms GA by 10% to 40% in terms of wasted area.

REFERENCES

[1] K. Shahookar and P. Mazumder, "VLSI Cell Placement Techniques." ACM Computer Survey, vol.23, no. 2, pp. 143–220, June 1991.

[2] E.Rashedi, H.Nezamabadi-pour, and S.Saryazdi, "GSA: A Gravitational Search Algorithm." Journal of Information of Science 179, 2232-2243, 2009.

[3] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "Binary Gravitational Search Algorithm." Springer Science + Business Media B.V. 2009

[4] H. R. Hassanzadeh, M. Rouhani, "A Multi-Objective Gravitational Search Algorithm." International Conference on Computational Intelligence, Communication Systems and Networks. 24, pp117-122, 2010.

[5] T. W. Manikas, M.H. Mickle, "A Genetic Algorithm for Mixed Macro and Standard Cell Placement." The 45th Midwest Symposium on Circuits and Systems, vol. 2, pp 115 - 118, vol.2, August 2002

[6] G. Nan, M. Li, D. Lin, J. Kou. Adaptive Simulated Annealing for Standard Cell Placement. Springer, 2005 Advances in Natural Computation Vol. 3612, pp 943-947, 2005

[7] A. R. Hosseinabadi, M. yazdanpanah and A. S. Rostami, A New Search Algorithm for Solving Symmetric Traveling Salesman Problem Based on Gravity, World Applied Sciences Journal 16 (10): pp 1387-1392, ISSN 1818-4952, 2012

[8] Gu W X, Li X T, Zhu L, "A gravitational search algorithm for flow shop schduling. CAAI Transactions on Intelligent Systems". 5(5): 411-418, 2010.

[9] J.P. Papa, A. Pagnin, S.A. Schellini, A. Spadotto. Feature selection through gravitational search algorithm. Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on, May 2011, pp: 2052 – 2055, 2011.

[10] W. Zhaoa, "Adaptive Image Enhancement based on Gravitational Search Algorithm." Procedia Engineering 15, pp. 3288 – 3292 Published by Elsevier Ltd. 2011.

[11] J. Xiao, Z. Cheng, "Theories and Applications DNA Sequences Optimization Based on Gravitational Search Algorithm for Reliable DNA computing." Sixth International Conference on Bio-Inspired Computing. IEEE Computer Society, 2011

[12] Rashedi E, Nezamabadi-pour H, Saryazdi S, "Filter modeling using gravitational search algorithm. Engineering." Applicaitons of Artifical Intelligence, 24, pp. 117-122, 2011

[13] M. Omar and J. Al-Neamy, "Hybrid Gravitational Search Algorithm and Genetic Algorithms for Automated Segmentation of Brain Tumors Using Feature_based Symmetric Ananlysis", (IJCSIS) International Journal of Computer Science and Information Security, Vol. 11, No. 5, May 2013.

[14] D. F. Wong, and C. L. Liu, A New Algorithm for Floorplan Design, Proc. DAC, pp.101–107,1986.

[15] J. P. Cohoon, S. U. Hedge, W. N. Martin, and D. S. Richards, “Distributed genetic algorithms for the floorplan design problem,” IEEE Trans. Computer-Aided Design, vol. 10, pp. 483–492, Apr. 1991.