VLSI Circuit Partitioning Using Ant Colony Optimization

Bharat Naresh Bansal

Assistant Professor, Department of Electronics

MIMIT Malout

bharat.mimit@gmail.com

Abstract

VLSI circuit partitioning problem is a physical design methodology, which divides a given circuit into segment abiding by some constraints and meeting certain objectives. The circuit portioning problem is NP hard. Ant colony optimization has been used to solve many computationally hard problems (NP hard problems). An evolutionary approach based on Ant colony optimization (ACO) is presented, which exploits the behavior of artificial ants modeled from real ants to solve the problem ACO has been used to solve many computationally hard problems(NP hard problems).Exiting conventional methods are unable to perform the required breakthrough in terms of complexity, time and cost.

***Keyword**: Circuit portioning, ACO, NP hard, Net list, Cut size, Pheromone, Node, etc.*

1. Introduction

With the advancements in VLSI technology the chip complexity is increasing which leads to more integration and increased in design sizes. The delay is also increases because of large area of chip is occupied by interconnects. As more number of components required to increase the functions in a circuit which results into a complex design and degradation in performance of a circuit. Circuit net list partitioning or circuit partitioning is an important step in VLSI physical design to solve these problems up to some extent. The main objectives of circuit net list partitioning include minimization of number of interconnections between the partitions, minimization of delay due to interconnections between partitions and ratio-cut minimization. In circuit partitioning, circuit is divided into n-parts to increase its performance by minimizing the critical path delay, power consumption and number of partition interconnection i.e. net cut. The partitioning process is NP-Hard. This type of problem is solved by heuristics. It can be grouped into two classes i.e. Constructive and Iterative. In the constructive mode the gates or components are not assigned to a partition. In case of iterative mode, the components are assigned to a partition.

Heuristic can also be further divided into two parts i.e. deterministic and non-deterministic . Earlier gives identical results, but non-deterministic gives different result each time when heuristic is run. The later heuristic give quality of solution in a shorter time than earlier one. The solution is evaluated keep in mind importance of both the objective and requirement constraints. In this paper biological inspired heuristic (ant colony) is used to solve such problem. The ant colony implemented is closely rooted at the biological and behavioral model of real social insects. It is non-deterministic heuristic and could be used as both constructive and iterative. The solution uses many ants of simple nature and limited memory requirements .The intelligence of the heuristic is not given by individual ants, but rather is expressed by the colony as a whole. The proposed metaheuristic is Ant Colony Optimization.

Applications of ACO algorithms fall into the two important classes of static and dynamic combinatorial optimization problems. Static problems are those whose topology and cost do not change while the problems are being solved.TSP is an example of static type in which city locations and intercity distances do not change during the algorithm's runtime. But in the dynamic problem the topology and costs can change while solutions are built, example of such type is routing in telecommunications networks, in which traffic patterns change all the time. Circuit partitioning is also similar type of problem. The early attempts to solve the circuit partitioning problem were based on the representation of the circuit as a graph G = (V,E), where V is a set of nodes (vertices) representing the fundamental components, such as gates, flip-flops, inputs and outputs and E is a set of edges representing nets present in the network. Each edge e £ E connects exactly two nodes Edge e_i of E is connected to exactly two nodes v_j,v_k of set V. Which is denoted by e_i=(v_j,v_k).Graph partitioning problems representing VLSI design problems usually involve separating the set of the graph nodes into disjoint subsets while optimizing some objective function defined on the graph vertices and edges. The objective functions associated with the graph partitioning problems usually treat these classes of edges in different ways.

Partitioning an appropriately designed geometric embedding of netlist, rather than a traditional graph representation, yields improve results as well as large speeds ups [20]. Better results has been taken by using Tabu search heuristic to guide an efficient interchange algorithm to reduce the number of the cut nets for K-partitions that are initially generated by a numerical eigen vector model [15]. Probabilistic method for circuit partitioning which gives good cutsets as compare to iterative as well as other techniques[20]. Neuro-memetic algorithm has been effective, especially in the case of a large number of cells when compared with other model. The algorithm has achieved very low minimum cut. In this, the circuit partitioning optimization is focused on finding an acceptable solution based on the delay, power, and cut-set cost [31]. Several researchers have contributed to solve the circuit partitioning by using different technique [21, 22, 30].

2. Need of Circuit Partitioning

The essence of netlist partitioning is to divide a system into clusters such that the number of inter-cluster connections is minimized. There are several reasons why partitioning has recently emerged as a critical step in many phases of VLSI systems synthesis. With rapid advancements in integration technology, it is possible to place a large number of logic gates on a single chip. Despite this fact, it may become necessary to partition a circuit because it is too large to be placed on a single chip or I/O pin limitations. The larger the gate count of the circuit, the larger the number of I/O pins associated with circuit. Due to this cost of packaging increases. So partitioning divides a system into smaller, more manageable components; the number of components corresponds to the interactions between the design sub-problems.

Finally, partitioning heuristics affect the layout area, improves autoroutbility, since it suggests reducing the wire congestion in any given layout region. All of these considerations motivate the development of netlist partitioning algorithms. The example of partitioning is shown in the figure 1.

3. Proposed Work

Ant colony optimization algorithm is used for partitioning the circuit. In this algorithm, mimics biological ants in finding their food and marking their own territory. In this work ant's model is followed as closely as possible. Many simple ants are used to accomplish the same task. Simple ants are oblivious of their environment and require very small memory (pheromone), which makes this model match closely that of the biological model. There is no central organization; simple rules applied by all ants will collectively form the new solution. The principal idea this approach is based on is self-organization.

Colonies of social insects composed of thousands of individuals, which have "cognitive abilities" that is better than the abilities of each of the individual members. This happens, as if the colony is ruled by the invisible hand of a central organizer. Each individual in the community works according to simple instinctive rules. They are totally unaware of the direction their entire society is heading. In fact, there is no direct connection between the individual behaviour and the colony direction as a whole.

Animat follow a set of local rules, which collectively create a solution to the circuit partitioning problem. The individual animats are unaware of the collective task they are accomplishing or the global state of the system. Local optimization techniques are also employed to assist the animats in achieving a good solution. The basic foundation of the algorithm is to consider each vertex in the graph or circuit as a location that can hold any animat. The animat can move around the graph by moving across edges to reach a new vertex. Each animat belongs to one of the species. However,

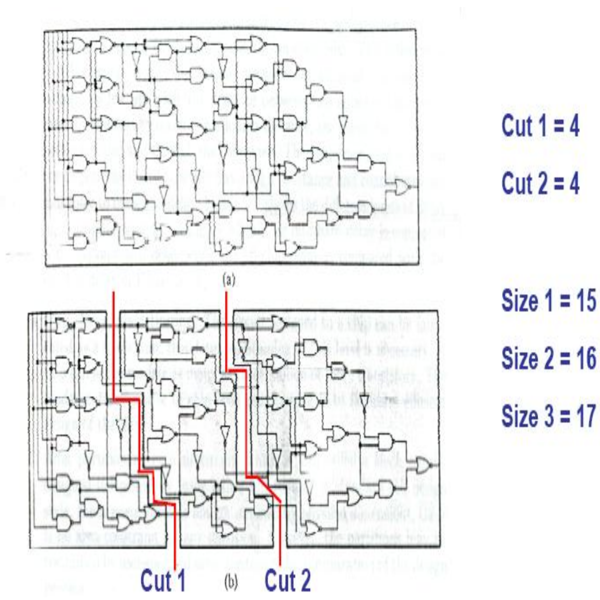

Figure 1. Circuit Partitioning

animats of all species follow the same rules. To start the algorithm, *n* animats are placed on the circuit. Their species and location are chosen randomly. At any point throughout the algorithm, the configuration of animats on the circuit constitutes a partitioning of the graph in the following way. Each vertex is considered to be colonized by one species. At a given time, it is said to be colonized by whichever species that has the greater number of animats on it. Any ties are recorded and the ties are broken in a random order by assigning the vertex to the species that result in a lower cut size. The set of all vertices colonized by species A constitutes A's colony and the vertices colonized by species B form B's colony and likewise others. In addition, each vertex can hold pheromone. The value of pheromone is updated after each tour of ant.

The idea of the algorithm is for each species of animats to form a colony consisting of a set of vertices that are highly connected to each other but highly disconnected from the other colony. The result should be two sets of vertices i.e. partitioning of the vertices into two sets that are highly connected amongst themselves, but have few edges going to the other sets. (i.e., a very small cut size).

4.1. Algorithm for Circuit Partitioning using ACO

1. First divide the circuit in to two parts and note down the cut set(interconnections between two parts).
2. Start the iteration from node1.
3. Move the ant to next node based on the probability and store the node.
4. If probability of two nodes equal than tie is there. Based on the cutest assign the nodes.
5. The ant can move or the tour length is equal to half of the total nodes.
6. Update the pheromone value after each movement of ant.
7. Store the tour after last iteration.
8. Evaporate pheromone value after each tour of ant.
9. Calculate the cut set.
10. Repeat same for different nodes .store all the cut set value.
11. Find the minimum cut set value from store value.

12. The above steps are repeated until the following condition satisfied.
    a) If new cut set >old cut set for half the nodes
    b) If new cut set =old cut set for half the nodes
    c) No. of times loop from step 1 to 11 occurred = nodes

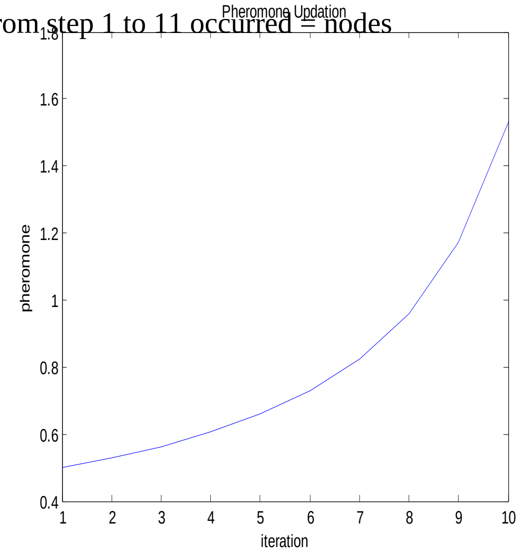

Figure 2. Update Pheromone Value after Every Movement

4.2. Mathematical Formulation

1. Take file as input and convert it into the matrix form.
2. Total ant or nodes= $\sum N_i$ Where i varies from 1 to n.
3. Now divide the circuit into two parts.
   Partition $\sum P_j$ and $\sum P_k$

   a) Where j varies from 1 to n_1 and k varies from $(n_1 + 1)$ to n.
   b) The number of nodes in both partition should be equal i.e. Balance criteria $\sum P_j = \sum P_k$ and $\sum N_i = \sum P_j + \sum P_k$ Calculate the gain of the circuit using the following formula.

4. Internal cost = I = (∑c(1:n1,1:n1))(∑(c(n1+1:n,n1+1:n))
   External cost = E = (∑c(1:n1,n1+1:n))(∑c(n1+1:n,1:n1))
   Gain = g = E-I
5. Start the movement of ants and initialize the parameters.
6. Take the first ant suppose from $\sum P_j$ and Move the ant based on the following probability.
   $p(v) = gv_c + pv_p + P_{min}$
   Where v_c is the number of vertices adjacent to v, g is the connectivity weight or gain and v_p is the amount of pheromone of the animat's species on vertex v, p is the pheromone weight and P_{min} ·is a fixed amount added to prevent any probabilities from being zero.
7. Update the pheromone value of the nodes by using following formula.
   Pheromone value = $\tau + \Delta\tau$
   Where, i) τ = Pheromone value = Previous pheromone value
   ii) Increment in pheromone value $\Delta\tau = (y)((\tau)(x))e^{-(\tau)(x)}$
   iii) Forage Pheromone X-Scale = X = 0.4
   iv) Forage Pheromone Y-Scale=Y = 1.5

Graph of above function shown in figure 2.

8. Evaporate the pheromone value based on the following formula.
   Pheromone value = $(\tau)*e^{-(\tau)(r)}$
   Where, τ = Pheromone value = Previous pheromone value
   Evaporation rate = r = 0.025

Graph of pheromone evaporation is shown in figure 3.

9. Now store the node in tour.
10. Note down the cutest.
11. Now repeat the above steps for different nodes and note down the cut set value.
12. Store the cut set values for different partition in a variable and find the minimum cut set value.

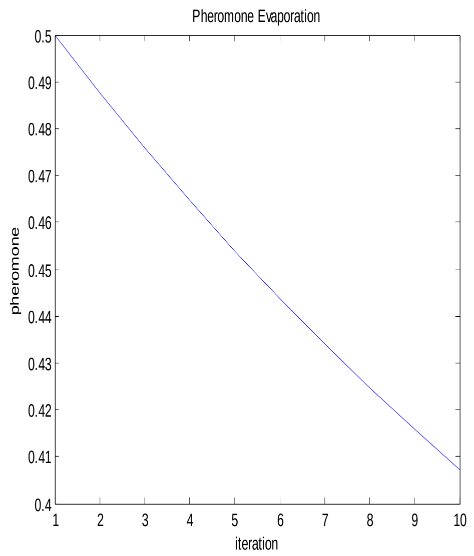

Figure 3. Evaporation of Pheromone Value

Repeat the above steps until following condition satisfied.

a) If new cut set >old cut set for half the nodes
b) If new cut set =old cut set for half the nodes
c) Number of times loop from step a to l occurred = nodes

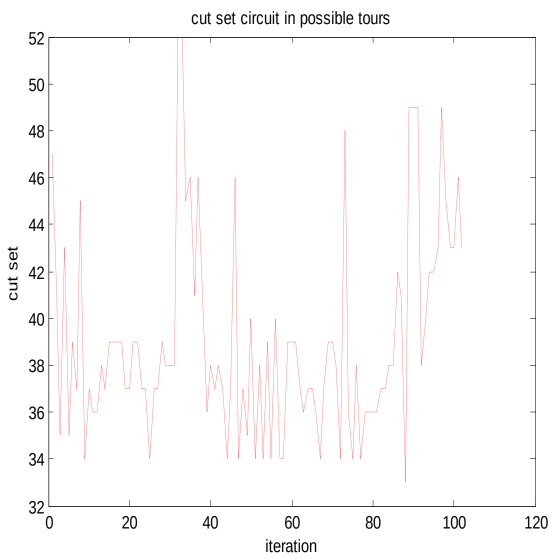

Figure 4. Net Cut Variations for Different Set

5. Results and Discussions

The Ant Colony heuristic produced very encouraging and comparative results for small and medium sized benchmark circuits. Table 1. below shows the results of test experiments for small and medium sized benchmark circuits. All results are obtained for balance criteria i.e. the number of components or gates in the partition should be equal, using this constrained we can compare this algorithm with other optimization algorithm. The figure 4. and 6. shows the variations of cut set of the benchmark circuit spp_N100_E103_R6_332 in different ant tours based on balance criteria which formed one set. It also shows that there is randomness in the algorithm which produced totally different result in each run. figure 7. shows that constant minimum cut set of the benchmark circuit spp_N100_E103_R6_332, when all the possible tours i.e. set is repeated for half the number of nodes times.

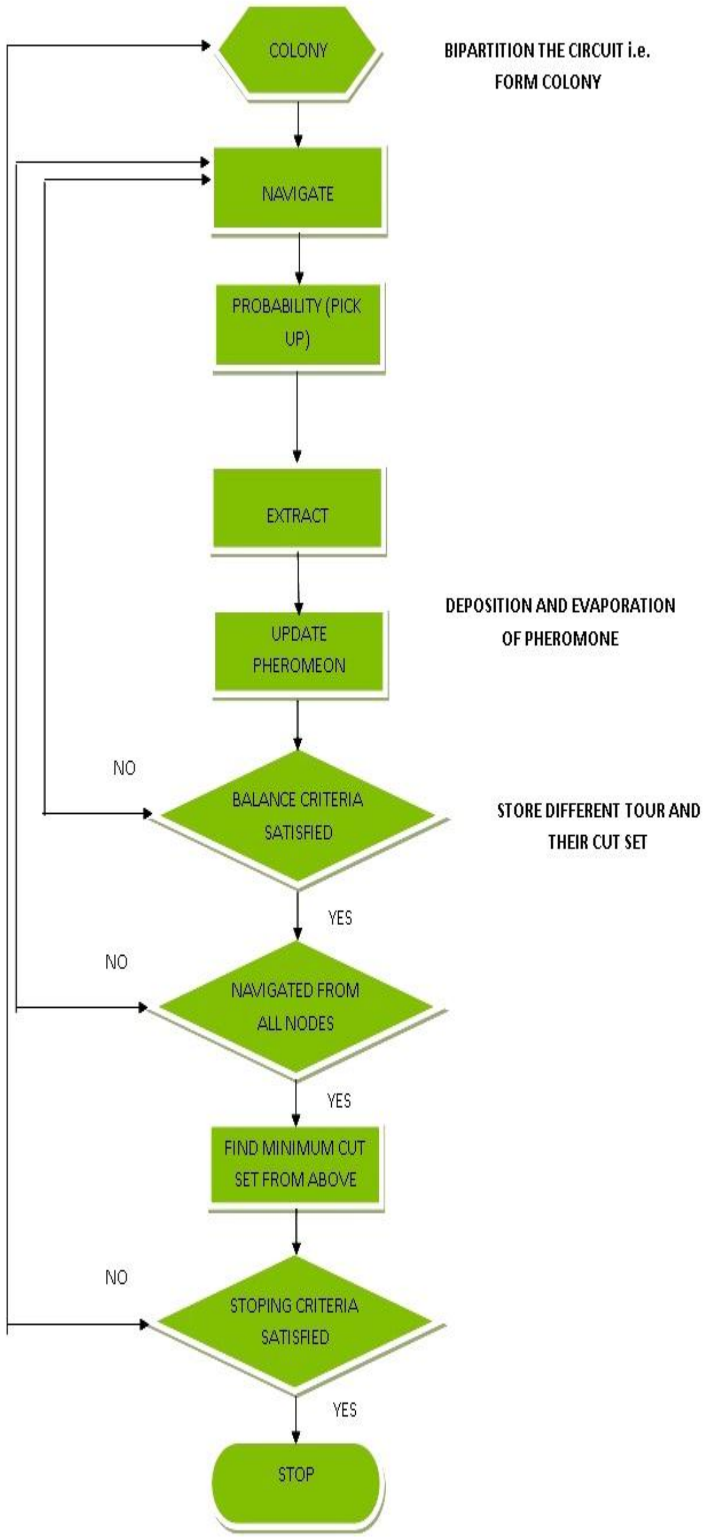

Figure 5. Flow Chart for Circuit Partitioning Using ACO

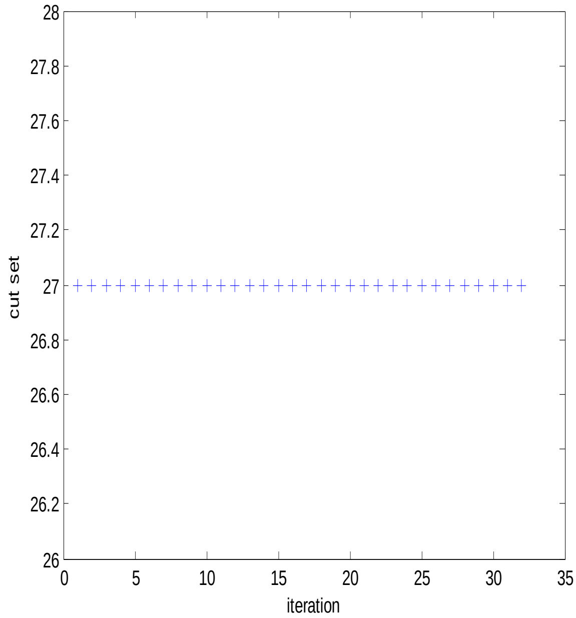

Figure 6. Net Cut Variations of Ants for Different Tour or in a Set

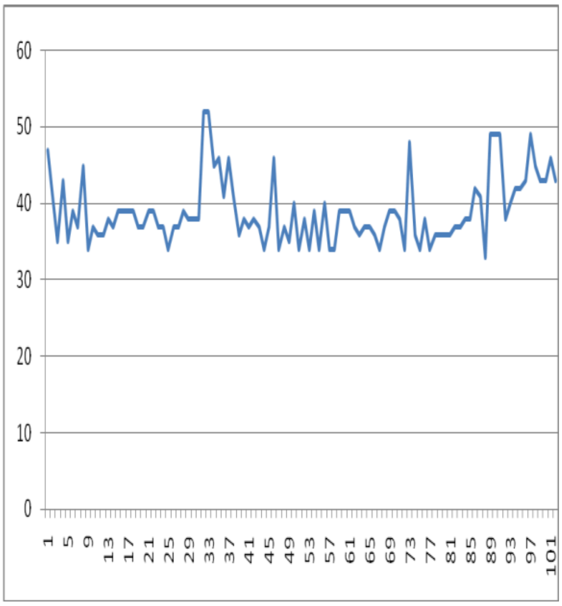

Figure 7. Net Cut variations for Different Set

6. Conclusion

The circuit partitioning problem is a well known NP hard problem. Various traditional algorithms are used to solve this problem. But due to complexity of different types of algorithms a better way is to use metaheuristic algorithms i.e. ACO. An algorithm using ACO techniques with local optimization was developed for solving the circuit partitioning problem. Animats are placed on a graph and move according to probability function. The emergent behavior of population following these rules, coupled with a local optimization, results in different partitions of the circuit with low cut size. Success of the algorithm is attributed to the fine mixture of simple definite steps and decisions based on random functions. This mixture of order and randomness is a characteristic of many processes in nature.

Table 1. Results of Partitioning on Benchmark

S. No.	File Name	Size	Net Cut
1.	spp_N8_E21_R1_18	8	4
2.	spp_N10_E7_R1_1025	10	5
3.	spp_N20_E21_R1_583	20	4
4.	spp_N30_E8_R1_25	30	3
5.	spp_N40_E40_R5_527	40	5
6.	spp_N45_E47_R3_268	45	3
7.	spp_N46_E45_R3_513	46	7
8.	spp_N50_E51_R3_599	50	4
9.	spp_N100_E103_R6_332	100	27
10.	spp_N189_E212_R10_199	189	28

Future Scope

In future scope more factors could be added to the move selection that would force the colonies away from each other. This could make the algorithm even more competitive when compared with other heuristic algorithms. From the theory side, researchers are trying either to extend the scope of existing theoretical results. From the experimental side, most of the current research is in the direction of increasing the number of problems that are successfully solved by ACO algorithms, including real-word, industrial applications. Currently, the great majority of problems attacked by ACO are static and well-defined combinatorial optimization problems, that is, problems for which all the necessary information is available and does not change during problem solution. For this kind of problems ACO algorithms must compete with very well established algorithms, often specialized for the given problem. Also, very often the role played by local search is extremely important to obtain good results. Although rather successful on these problems, it is believed that ACO algorithms will really evident their strength when they will be systematically applied to " ill-structured" problems for which it is not clear how to apply local search, or to highly dynamic domains with only local information available. A first step in this direction has already been done with the application to telecommunications networks routing, but more research is necessary.

References

1. S. C. Johnson, "Hierarchical Clustering Schemes", Psychometrika., vol. 32, no. 3, **(1967)**, pp. 241-254.

2. E. Lawler, K. Levitt, and J. Turne, “Module Clustering to Minimize Delay in Digital Networks”, IEEE Transacations on Computer-Aided Design., **(1969)**, pp. 47-57.
3. B. W. Kernighan and S. Lin, “An Efficient Heuristic Procedure for Partitioning Graphs”, The Bell System Technical Journal., vol. 29, no. 2, **(1970)**, pp. 291-307.
4. D. G. Schweikert and B. W. Kernighan, “A Proper Model for the Partitioning of Electrical Circuit” In 9th Annual Design Automation Workshop, **(1972)**, pp. 57-62.
5. C. M. Fiduccia and R. M. Mattheyses, “A Linear-Time Heuristic for Improving Network Partitions”, Proceedings of the 19th IEEE Design Automation Conference, **(1982)**, pp. 175-181.
6. B. Krishnamurthy, “An Improved Min-Cut Algorithm for Partitioning VLSI Networks”, IEEE Transactions on Computers., vol. 33, **(1984)**, pp. 438-446.
7. T. K. Ng, J. Oldfield and V. Pitchumani, “Improvements of a Mincut Partition Algorithm”, In IEEE/ACM International Conference on Computer Aided Design, **(1987)**, pp. 470-473.
8. J. De Gruijter and A. B. McBratney, “A Modified Fuzzy K-Means Method for Predictive Classification”, In Proceedings of the First Conference of the International Federation of Classification Societies (IFCS), **(1988)**.
9. C. Sechen and D. Cheng, “An improved Objective Function for Mincut Circuit Partitioning”, Proceedings of Int. Conf. on Computer-Aided Design, **(1988)**, pp. 502-505.
10. C. Kring and A. R. Newton, “A Cell-Replicating Approach to Mincut-Based Circuit Partitioning”, In IEEE ICCAD-91, **(1991)**, pp. 2-5.
11. L. Hagen and A. Kahng, “New Spectral Methods for Ratio Cut Partitioning and Clustering” IEEE Trans. CAD., vol. 11, no. 9, **(1992)**, pp. 1074-1085.
12. J. Hwang and A. El Gamal, “Optimal Replication for Min-Cut Partitioning”, In IEEE/ACM International Conference on Computer Aided Design, **(1992)**, pp. 432-435.
13. J. Cong and M. Smith, “A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design”, In 30th DAC, ACM/IEEE, **(1993)**, pp. 755-760.
14. H. Shin and C. Kim, “A Simple Yet Effective Technique for Partitioning” IEEE Transaction on VLSI, **(1993)**, pp. 380-386.
15. S. Areibi and A. Vannelli, “A Combined Eigenvector Tabu Search Approach For Circuit Partitioning” In Proceedings of the Custom Integrated Circuits Conference (San Diego), **(1993)**, pp. 9.7.1 - 9.7.4.
16. C. Ball, P. Kraus, and D. Mlynski, “Fuzzy Partitioning Applied to VLSI Floorplanning and Placement” Proceedings IEEE Intl. Symp. Circuits and Systems, **(1994)**, pp. 177-180.
17. S. Areibi and A. Vannelli, “Advanced Search Techniques for Circuit Partitioning, in Quadratic Assignment and Related Problems”, DIMACS Series in Discrete Mathematics and Theoretical Computer Science edited by P. Pardalos and H. Wolkowicz., vol. 16, **(1994)**, pp. 77-99.
18. S. Barnard and H. Simon, “A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems”, Concurrency: Practice & Experience., vol. 6, no. 2, **(1994)**, pp. 101-117.
19. D.J-H. Huang and A.B. Kahng, “On Implementation Choises for Iterative Improvement Partitioning Algorithms”, In Proceedings of Euro-DAC'95, **(1995)**, pp. 144-149.
20. C. J. Alpert and A. B. Kahng, “A Hybrid Multilevel/Genetic Approach for Circuit Partitioning”, Physical Design Workshop, **(1996)**, pp. 100-105.
21. Shantanu Dutt and Wenyong Deng, “A Probability-Based Approach to VLSI Circuit Partitioning”, In 33th DAC, ACM/IEEE, **(1996)**, pp. 644-649.
22. Slawomir Koziel and Wladyslaw Szczesniak, “Evolutionary Algorithm for Electronic Systems Partitioning and its Applications in VLSI Design”, IEEE Computing., **(1999)**, pp. 1411-1414.
23. M.Dorigo and G.Di Caro, “The ant colony optimization meta-heuristic”, In New Ideas in optimization”, Eds.London, UK:McGraw Hill, **(1999)**, pp. 11-32.
24. D. Angus and Hendtlass T., “Ant Colony Optimization Applied to Dynamically Changing Problem”, Development in Applied Artificial Intelligence LNA12358., **(2002)**, pp. 618-627.
25. Peter Koro, Jurij and Borut Robi, “A Multilevel Ant Colony Optimization Algorithm for Mesh Partitioninig”,International Journal of Pure and Applied Mathematics., vol.5, **(2003)**, pp. 143-159.
26. Marco Dorigo and Thomas Stutzle, “Ant Colony Optimization”, ISBN-10:0-262-04219-3,ISBN-13:978-0-262-04219-2, **(2004).**
27. Gang Wang, Wenrui Gong and Ryan Kastner “System Level Partitioning for Programmable Platforms Using the Ant Colony Optimization”, International Workshop on Logic and Synthesis (IWLS). **(2004).**
28. Borut Robic,Peter Korošec, and Jurij Silc, “Handbook of Bioinspired Algorithms and Applications”Chapter 19th of the book, “Ant Colonies and the Mesh-Partitioning Problem”, ISBN-10: 1-58488-475-4 ,ISBN-13: 978-1-58488-475-0 , **(2006)**, p.p 285-301.
29. Sumithra Devi K A, Dr. Annamma Abraham, Banashree N P, “An Evolutionary Time- Series Model for Partitioning A Circuit Pertaining to VLSI Design Using Neuro Memetic Algorithm”, The Fourth IASTED International Conference on Circuits, Signals, and Systems,San Francisco, USA, **(2006),** pp. 325-329.
30. K. A. Sumitra Devi, N. P. Banashree, and Annamma Abraham, “Comparative Study of Evolutionary Model and Clustering Methods in Circuit Partitioning Pertaining to VLSI Design” In Proceedings of World Academy of Science,Engineering and Technology, **(2007)**, vol. 20 ISSN 1307-6884.