An Algorithm for Min Cut Partitioning for Digital Circuit Layout based on Evolutionary Approach

Maninder Kaur
School of Mathematics and Computer Application,
Thapar University, Patiala, INDIA
Email: manindersohal@thapar.edu

Kawaljeet Singh
Director, University Computer Centre, Punjabi University
Patiala, INDIA
Email:singhkawaljeet@rediffmail.com

***Abstract*—The paper presents a new algorithm for circuit partitioning that exploits some of the latest innovations of evolutionary approaches. The proposed algorithm based on swarm intelligence technique hybrids discrete version of artificial bee colony algorithm with Inver over operator along with simulated annealing technique for local improvement of solution. The algorithm is tested on small UCLA instances and the proposed algorithm outperforms in terms of average runtime for average net cut in comparison to UCLA branch and bound partition.**

***Keywords*- Artificial Bee Colony algorithm, Simulated Annealing, Inver-over operator**

I. INTRODUCTION

In the combinatorial sense, the digital circuit layout problem is a constrained optimization problem. Practically, all aspects of the layout problem as a whole are intractable [11, 12].Many heuristic methods are incorporated to solve very large problems. One of these methods is to divide a circuit hierarchically into parts with divide and conquer technique .This paper focuses on the problem of min- cut weighted circuit bipartition .Other than being the first phase in physical digital circuit design; circuit partitioning is of important use in placement, floor planning, and other layout problems.

Nature is inspiring researchers to develop models for solving their problems. The Circuit Partitioning problem being a classic NP-hard problem, various evolutionary approaches like genetic algorithm [2,14,15], ant colony system [1] , memetic algorithm[16] ,simulated evolution algorithm[10] honey bee algorithm[7] and particle swarm optimization[9] have been proposed to solve the circuit partitioning based on graph partitioning technique ,which are used to provide solutions of high quality but not necessarily optimal. This paper presents a new hybrid algorithm, which is based on the concepts of the artificial bee colony (ABC) with Inver-Over operator for neighborhood search and local improvement strategy using simulated annealing technique.

II. PROBLEM FORMULATION

A circuit is represented as a netlist hypergraph *H(V,E)* where *V={v1,v2……vn}* is set of vertices or modules and E is set of edges/ nets which is a subset of V. Let *A (v)* denote the total area of $v \in V$ and let $A(S)=\sum_{v \in s} A(v)$ denote the area of a subset $S \subset V$.

The cut of a bipartition, Cut (*p*) is the number of nets which contain modules in both *X* and *Y* i.e

$$Cut(P) = |\{e|e \cap X \neq \emptyset,\ e \cap Y \neq \emptyset\}|. \quad (1)$$

An optimal min cut area-balanced/vertex weighted bipartition *P={X, Y}* is a pair of disjoint sets *X* and *Y* such that *X U Y=V* with minimum cut (P) subject to the constraint

$A(V)(1-r)/2 \leq A(X), A(Y) \leq A(V)(1+r)/2$ with *r* as user defined tolerance factor.

III. THE PROPOSED ALGORITHM

In this work The HABCSACP (Hybrid Artificial Bee Colony Optimization with Simulated Annealing for Circuit Partitioning) is proposed .This hybrid algorithm uses Artificial Bee Colony algorithm[3,5] which has good exploration and exploitation capabilities in searching optimal solution with Inver-over operator[13] for neighbor search and simulated annealing for local improvement of solution

In the algorithm a group of bees is created during initial stage. The count of number of Employed bees ($N_{employed}$) and Onlooker bees ($N_{onlooker}$) , both is set equal to the population size.

Initialize pop_size - Number of solutions in the population
maxCycle - Total no. of iterations
limit - Total number of trials after which the solution is rejected,
lchrom - Total number of modules/vertices of circuit i.e. length of solution

Pseudo code of the Proposed Algorithm

```
Intialise the parameters :pop_size, maxcycle, limit, lchrom
 Set N_employed =N_onlooker= pop_size
Set limit, maxcycle
Initialize (pop_size,lchrom);
for iter=1:maxcycle
        Sendemployedbees(pop_size,lchrom);
        Calculateprobabilites(pop_size);
        Sendonlooker(pop_size,lchrom);
[Globalparam,globalmin]=Memorizebestsource(pop_size);
Local_improve_SA(Globalparam,globalmin);
SendScoutBees(pop_size,limit,lchrom);
End
Write : Globalparam, Globalmin  //resultant solution
```

End.

Step 1: *Initialize (pop_size,lchrom)* : Randomly generate an initial population P of size pop_size with set of feasible solutions(i.e. balanced partition w.r.t weight of vertices) .Read the input files and convert it into netlist format. Calculate the fitness value of each solution in the population using the net cut evaluation mechanism.

For a net cut evaluation a multiword mask of size of the chromosome is pre computed for each net .If a cell is connected to net, the corresponding bit position is set

$$M_{ij} = \begin{cases} 1 & if \ \ Cj \in Ni \\ 0 & otherwise \end{cases} \quad (2)$$

Where C_j is the j^{th} cell in order

M_{ij} = mask for net N_i and is the *jth* bit position of M_i.

The value of CM_i and $\hat{C}M_i$ is evaluated. If both values are nonzero i.e. net is present in both partitions, hence a cut. Otherwise no cut

Step 2: Repeat the following steps for specified number of iterations i.e. maxcycle

Step 2.1: *Sendemployedbees(pop_size,lchrom) :* For each solution in the population produce a modification on the one existing. If the fitness value of new solution is higher than that of the previous one then memorizes the new solution and forget the old one. Otherwise keep the position the previous one

For each solution S_i for $i \in [1: N_{employed}]$
SN_i =Inver-over operator (S_i) //Generate neighbourhood solution SN_i
Calculate the fitness value of SN_i .
If fitness(SN_i)>fitness(S_i)
S_i=SN_i
S_i.trial=0
Else
S_i.trial= S_i.trial+1
End of if statement
End of for statement

Every employed bee in the population chooses a neighborhood solution. The neighborhood solution is generated by Inver-over operator [ ] which has a strong selective pressure with an adaptive operator. The features of this adaptive operator are the number of inversions applied to a single individual and the segment to be inverted is determined by another (randomly selected) individual.

The probability *p* of generating random inversion, and the number of iterations in the termination condition

Pseudocode of Inver-over operator

Sol= S_i
select (randomly) a vertex v from Sol
While (1)
if (rand() ≤p) then select the vertex v_0 from the remaining vertices in Sol
else select (randomly) an individual from P, assign to v_0 the 'next' vertex to the vertex v
in the selected individual
end of if statement

if (the next vertex or the previous vertex of vertex v in Sol is v_0) then exit from loop

end of if statement

inverse the section from the next vertex of vertex v to the city v_0 in Sol
v = v_0
if (fitness(Sol) ≤ fitness(S_i)) then S_i = Sol
End of while

Step 2.2 *Calculateprobabilites(pop_size);* The probability value is calculated for each solution is calculated using the following formula

$$S_i.p = \frac{fi}{\sum_{i=1}^{pop\_size} fi} \quad \textbf{(3)}$$

where pop_size -is the population size and

f_i is the fitness value of i^{th} solution

Step 2.3 *SendOnlookerbees(pop_size,lchrom):* At the third stage, an onlooker prefers a solution depending on the probability value of the solution.

For each solution S_i for $i \in [1: N_{employed}]$
If(Rand() < S_i.p)
then
SN_i =Inverover operator (S_i) //Generate neighbourhood solution SN_i.
Calculate the fitness value of SN_i .
If fitness(SN_i)>fitness(S_i)
S_i=SN_i
S_i.trial=0
Else
S_i.trial= S_i.trial+1
End of if statement
End of for statement

Step 2.4 *Local_improve_SA(Globalparam,globalmin)* Simulated annealing being a simulated annealing algorithm proposed by Kirkpatrick et al. [8] is incorporated into the algorithm where the resultant solution is further improved by finding nearby solutions. The simulated annealing algorithm gives optimal results for circuit partitioning by locally improving the solution [4].

Step 2.5 *SendScoutBees(pop_size,limit,lchrom):* If the solution trial reaches the limit value then it is abandoned and new solution is generated randomly and replaced with the abandoned one.

index = 1;
For each solution Si for $i \in [2: pop\_size]$
if (Si.trial > Sindex. trial)
index= i;
end of if statement
end of for statement

```
if  (Sindex. trial >=limit)
        popnext=generate_new_solution (index,lchrom);
   end of if statement
end of for statement
```

After specified number of iterations the final solution is printed and runtime is calculated for multiple number of partitioning instance groups in each size range of test circuit instances.

IV. RESULTS AND DISCUSSIONS

The algorithm was implemented in the MATLAB (version 9) on an Intel Core i5 (2.60 GHz) machine with 4 GB memory. The performance of the proposed algorithm is tested on UCLA small circuit partitioning instances [6] generated by the top-down partitioning-based placement process employed by the UCLA Capo placer.

TABLE I. COMPARISON OF AVERAGE RESULTS FOR THE PROPOSED ALGORITHM AND UCLA BRANCH AND BOUND PARTITIONER ON NUMEROUS PARTITIONING INSTANCE GROUPS.

S.No	SPP Circuit–series	No. of Nodes	No.of files	UCLA Branch-and-Bound Partitioner		Proposed algorithm	
				Average Cut	Average Runtime	Average cut	Average Runtime
1	spp-N10 series	10	483	4.1	0.00035	4.03	0.00030
2	spp-N15 series	15	184	5.4	0.00063	5.17	0.00059
3	spp-N20 series	20	121	7.2	0.00160	7.05	0.00152
4	spp-N25 series	25	107	7.6	0.00438	7.78	0.00459
5	spp-N30 series	30	52	8.0	0.00984	7.75	0.00632
6	spp-N35 series	35	31	10.4	0.06092	10.5	0.05999
7	spp-N40 series	40	41	8.4	0.21786	8.4	0.18672
8	spp-N45 series	45	28	11.2	0.68483	10.75	0.6500
9	spp-N50 series	50	24	10.5	3.93902	10.25	3.7852
10	spp-N55 series	55	20	11.7	236.48801	11.6	232.4723
11	spp-N60 series	60	9	11.6	31.75144	10.8	30.7864

As seen from Table I, average results obtained by the proposed algorithm are consistently better than these obtained by UCLA branch and bound partitioner for all partitioning instances over different size ranges.

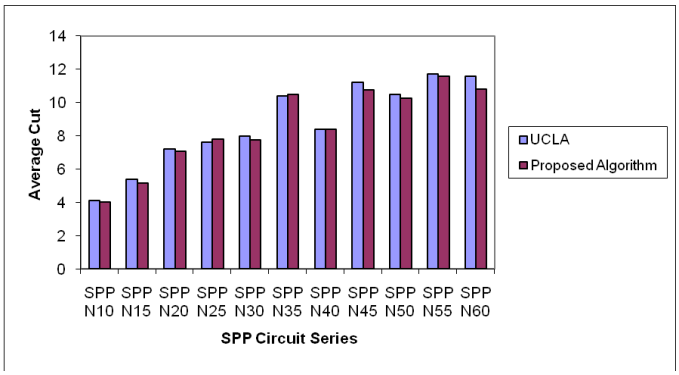

Figure 1. SPP combinational circuits Average Cut time

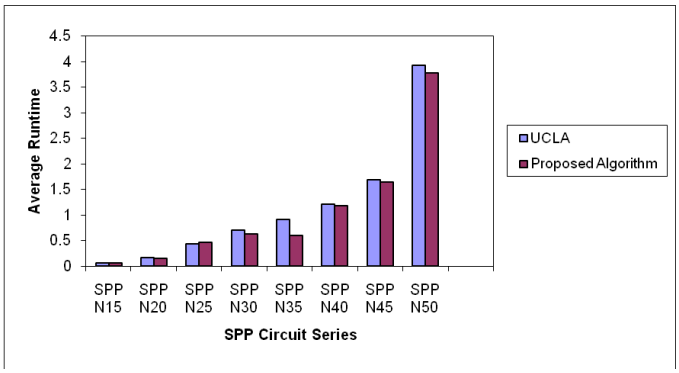

Figure 2. SPP combinational circuits Average Run time

These circuits are given in multiple number of partitioning instance groups in each size range as shown in figure 1 and 2. The circuit net lists are in the nodes/nets/weights format. The average results from the proposed algorithm have been compared with those obtained by the UCLA branch and bound partition

V. CONCLUSIONS

In this paper, a new hybrid algorithm based on evolutionary approach is proposed for circuit partitioning problem. The proposed algorithm incorporates InverOver operator in artificial bee colony algorithm along with simulated annealing for further improvement of the resultant solution. The algorithm is tested on 11 spp- circuit series of UCLA small circuit partitioning instances given on the MARCO GSRC VLSI CAD bookshelf website each further having set of circuits .The experimental results showed that the proposed algorithm give better and consistent results than UCLA branch and bound partitioner. Results obtained show the versatility of the proposed method in solving non polynomial hard problem of circuit net list partitioning.

REFERENCES

[1] A.E. Langham and P.W. Grant. Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies. In D. Floreano, J-D. Nicoud, and F. Mondada, editors, In *Proc. 5th European Conference on Artificial Life, ECAL'99*, volume 1674 of *LNCS*, pages 621-625, Swiss Federal Institute of Technology, Lausanne, September 1999. Springer. (Research Report: CSR 13-99).

[2] C. J. Alpert, L. W. Hagen, and A. B. Kahng, "A hybrid multilevel/genetic approach for circuit partitioning," in *Proc. ACM/SIGDA Physical Design Workshop*, 1996, pp. 100–105.

[3] D. Karaboga, An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical ReportTR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[4] D. Kolar, J. Divokovic Puksec and Ivan Branica, " VLSI Circuit partitioning using Simulated annealing Algorithm", IEEE Melecon, Dubrovnik, Croatia,May 12-15,2004.

[5] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (2008), pp.687-697, 2008.

[6] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Partitioning/

[7] James D. McCaffrey,"Graph Partitioning using a Simulated Bee Colony Algorithm", Proceedings of the 12th IEEE International Conference on Information Reuse and Integration, August 2011, pp. 400-405.

[8] Kirkpatrick S., Gelatt C. Jr. and Vecchi M. Optimization by Simulated Annealing. Science, 220(4598):671-680, 1983.

[9] Lingyu Sun, Ming Leng: An Effective Refinement Algorithm Based on Swarm Intelligence for Graph Bipartitioning. ESCAPE 2007: 60-69

[10] M. Sait, Aiman H. El-Maleh, Rush H. Al-Abuji: Simulated evolution algorithm for multiobjective VLSI netlist bi-partitioning. ISCAS (5) 2003: 457-460.

[11] N. Sherwani, Algorithms for VLSI Physical Design and Automation, 3rd ed., Springer (India) Private Limited, New Delhi, 2005

[12] Sadiq M. Sait and Habib Youssef. VLSI Physical Design Automation: Theory and Practice. McCraw-Hill Book Company, Europe, 1995.

[13] T.Guo and Z.Michalewize. Inver-Over operator for the TSP. In Parallel Problem Solving from Nature(1998)

[14] T. N. Bui and B. R. Moon, "A fast and stable hybrid genetic algorithm for the ratio-cut partitioning problem on hypergraphs," in *Proceedings of the ACM/IEEE Design Automation Conference*, 1994, pp. 664–669.

[15] T. N. Bui, B. R. Moon, "Genetic Algorithm and Graph Partitioning". In IEEE Transactions on Computers, Vol.45, No.7, July 1996, pp. 841-855.

[16] U. Benlic and J.-K. Hao. An Effective Multilevel Memetic Algorithm for Balanced Graph Partitioning. In *Proc. 22nd IEEE Intl Conf. Tools with Artificial Intelligence*, pages 121-128, 2010