Ant Colony Optimization and its Application to Boolean Satisfiability for Digital VLSI Circuits

Rajamani Sethuram rajamani@caip.rutgers.edu
Manish Parashar parashar@caip.rutgers.edu
Electrical and Computer Engineering Dept., Rutgers University
Piscataway, NJ 08854, USA

Abstract

Ant Colony Optimization (ACO) [8] is a non-deterministic algorithm framework that mimics the foraging behavior of ants to solve difficult optimization problems. Several researchers have successfully applied ACO framework in different fields of engineering, but never in VLSI testing. In this paper, we first describe the basics of the ACO framework and ways to formulate different optimization problems within an ACO framework. We then present our own ACO algorithm to simultaneously solve multiple Boolean SAT instances for digital VLSI circuits. Experiments conducted on scanned versions of ISCAS'89 benchmark circuits produced astonishing results. ACO framework for Boolean Satisifiability was found 200 times faster than spectral meta-heuristics [36] run in combinational mode.

ACO framework has proven to be a promising optimization technique in large number of other fields. Since ACO can be used to solve different types of optimization and search problems, we believe that the concepts presented in this paper can open the gates for researchers solving different optimization problems that exist in VLSI testing more efficiently.

1 Introduction

Social insects such as ants, bees, wasps, and termites exhibit distributed control, local interaction and communication that help themselves organize. Grasse [17] first studied the behavior of a kind of termites during the construction of their nests and noticed that the coordinated behavior of these insects during the construction process is influenced by the structure of the construction themselves. They use the environment as a medium to represent their past behavior that in turn influences their future behavior. Such a process is termed *self catalytic process* i.e. the more a process occurs, the higher its chances of occuring in the future.

Also, ants have the ability to find the shortest path while searching for food. Ants indirectly communicate with other ants by leaving a trail of chemical substance, called *pheromone*, that guides other ants to take the shortest path. Figure 1 shows a colony of ants searching for food. In Figure 1(a), ants start exploring all available paths while searching for food. Figure 1(b) shows that eventually most of the ants choose the shortest path. This indirect form of communication, that enables ants to find shortest path, is called *stigmergy* [17, 34].

Inspired by the self catalytic and stigmergytical ant behavior, Dorigo [10, 11] and his colleagues developed ant colony optimization (ACO) *meta-heuristic* to solve difficult combinatorial optimization problems. The term meta-heuristic means a general-purpose heuristic method designed to guide an underlying problem-specific heuristic

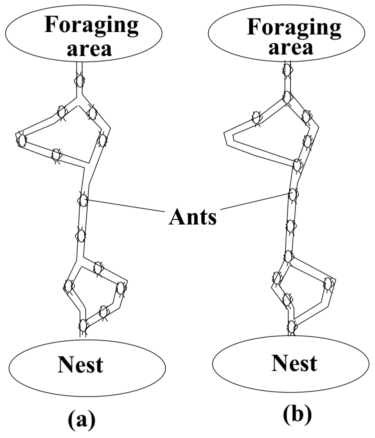

Figure 1: Foraging Ants. a) Ants start exploring, b) Eventually most of the ants choose the shortest path

(e.g. local search problem) toward promising regions of the search space containing high-quality solution. Their algorithm mimics the foraging behavior of ants to solve any optimization or search problem. They first applied ACO algorithm to solve the traveling salesman problem and showed that their solution converged very quickly to the optimal solution. Since then, researchers working on a wide variety of areas have successfully extended and applied ACO algorithm to solve several optimization and search problems. Table 1 gives a subset of recent work that uses ACO to solve a wide variety of problem in different fields of engineering and sciences.

Table 1: Application of ACO

Problem	Citation
Network Routing	[30, 31]
Quadratic Assignment Problem	[14]
Vehicle Routing	[13]
Mobile Network	[12, 18]
Continuous Space Optimization	[15]
Software Testing	[22]
Macrocell Overlap Removal	[1]
DNA Fragment Assembly	[24]

There is a large number of optimization problems in VLSI testing (see Table 2) that have been solved using different meta-heuristics, but never using ACO. This paper describes the basics of ACO and then presents an ACO frame-

Authorized licensed use limited to: IEEE Xplore. Downloaded on November 21, 2008 at 11:41 from IEEE Xplore. Restrictions apply.

work for solving multiple Boolean SAT instances for digital VLSI circuits. We strongly believe that the concepts described in this paper can open the gates for researchers to solve a variety of optimization problems in testing as well as help to come up with more optimization problems for ACO application in the Œeld of VLSI testing. Our experiments with ISCAS'89 benchmark shows that ACO algorithm is up to 200 times faster than spectral meta-heuristics [36].

Table 2: Optimization Problems in VLSI Testing

Problem	Ref.	Technique Used
Sequential Testing	[19]	Genetic algorithm
Partial Scan	[20]	
Scan Chain Partitioning and Re-ordering	[2]	
Test Bus Architecture to Minimize Test Time	[29]	
Combinational ATPG	[4, 5]	Neural Network
Delay Testing	[6]	Energy Minimization
TAM Designing	[3]	Integer Linear Programming
Optimal Compacted Test Set	[32]	
Scan Chain Ordering	[22]	Simulated Annealing
SOC Test Scheduling	[37]	
Sequential Testing	[7, 36]	Spectral Transform
BIST Response Compaction	[21]	

The rest of the paper is organized as follows. Section 2 describes the basics of ACO. In Section 3, we describe our novel ACO framework to simultaneously solve multiple Boolean SAT instances. In Section 4 we present results and Section 5 presents conclusion and future work.

2 Basics of ACO

In the ACO meta-heuristic, a colony of *artiŒial ants* (also called ants, in short) cooperates in Œnding global good solutions to difŒult optimization problems. ArtiŒial ants are an abstraction of real ant behavior. They are sometimes enriched with some capabilities which do not Œnd a natural counterpart. These capabilities make the artiŒial ants more effective and efŒient for solving a particular problem. Ants cooperate by means of the information they concurrently read/write on the problem's state they visit. As described earlier, natural ants leave pheromone trails while moving. These pheromone trails evaporate over time, which allows ant to forget its prior state. To mimic this behavior in an ACO framework, artiŒial ants leave numeric information called *artiŒial pheromone trail* (APT). APT left by an artiŒial ant represents the current state and/or the history of that ant. This stigmergetic form of communication using APT among individual ants guide all ants to obtain the global optimal solution efŒiently.

ArtiŒial ants move from one problem state to adjacent problem state using probabilistic decision policy, which is a function of APT. Exact deŒnition of state, adjacency, and decision policy are problem speciŒc. An important feature of the decision policy is that it uses only local information and does not use any lookahead mechanism to make a decision.

2.1 ACO for Traveling Salesman

We now describe an ACO framework (see Figure 2) for solving traveling salesman problem [9]. Initially, m ants are generated that collectively works to build a shortest closed tour of the given graph, where m=n and n is the total number of cities. When the kth ant, A_k, traverses an arc (i, j), it deposits a quantity of pheromone $\Delta\tau_{ij}^k(t)$ on this arc that is given by

$$\begin{aligned}\Delta\tau_{ij}^k(t) &= 1/L_k(t) \;\; if(i,j) \in T^k(t) \\ &= 0 \;\; otherwise\end{aligned} \tag{1}$$

where $T^k(t)$ is ant A_k's tour at iteration t, and $L^k(t)$ is its length. Thus, $\Delta\tau_{ij}^k(t)$ represents how well the ant has performed. The pheromone trail strength $\tau_{ij}(t)$ is updated using the equation given below

$$\tau_{ij}(t) \;\; \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2}$$

where $\Delta\tau_{ij}(t)$ = $\sum_{k=1}^{m}$ $\Delta\tau_{ij}^k(t)$ and ρ $\in$ (0, 1] is the pheromone trail decay co-efŒient.

Each ant A_k also maintains a *tabu list* denoted by Tb_k, which is the set of cities that ant A_k has visited. All ants use the tabu list to make sure that a particular city is not visited twice. If an ant A_k is located at city i, then the probability that it will move to city j is given by

$$p_{ij}^k \;\; = \frac{a_{ij}(t)}{\sum a_{il}(t)} \tag{3}$$

where the summation in the denominator is done for the set of all nodes l in the neighborhood of node i that the ant A_k is not visited yet. Here $a_{ij}(t)$ is deŒned as

$$a_{ij}(t) \;\; = \frac{[\tau_{ij}(t)]^\alpha[\eta_{ij}]^\beta}{\sum[\tau_{il}(t)]^\alpha[\eta_{il}]^\beta} \tag{4}$$

where the summation in the denominator is done for set of all nodes l in neighborhood of node i, α and β are two parameters that control the relative weight of pheromone trail and are set to 1 and 5 respectively, η_{ij}=1/d_{ij} where d_{ij} represents the distance between city i and city j. The overall algorithm for the ACO framework is given in Figure 2. The

```
AcoForTSP(m)
1. for i ← 1 to 3000
2.     Generate m ants, one for each city
3.     for j ← 1 to m
4.         for k ← 1 to m
5.             Move ant A_k to the next
                   location using equation 3
6.     Save the best soln. generated in this iteration
7.     Update pheromone strength using equation 2
8. Print the best solution obtained
```

Figure 2: ACO Algorithm for Traveling Salesman

simple ACO framework presented above for the traveling salesman problem exhibited quick convergence to good solutions. This stimulated the researchers to use ACO algorithm for problems whose state space is dynamic.

2.2 ACO for Dynamic State Space

Routing packets in mobile ad hoc network is a dynamic state space problem because the network topology and parameters in a mobile ad hoc network (MANET) changes dynamically. Ducatelle *et al.* [12] proposed a multi-path

Authorized licensed use limited to: IEEE Xplore. Downloaded on November 21, 2008 at 11:41 from IEEE Xplore. Restrictions apply.

routing algorithm called AntHocNet that can dynamically learn the changing network parameters and efŒciently route data packets from one node to another node in a MANET. It consists of *reactive* and *proactive components*. The reactive component (RC) is invoked only when there is a request for transmitting data from one node to another. In that case RC launches *reactive forward ants* starting from the source node to Œnd multiple paths to the destination and *backward ants* return to the source node to set up the path. A proactive component is invoked while the data transmission is going on during which the paths are monitored, maintained and improved proactively. From the ACO viewpoint, this work presented the following key ideas: a) using two different phases viz. the reactive phase and the proactive phase in the top level ACO algorithm, b) using a combination of both deterministic (for detecting and updating link failure) and probabilistic approaches (ant movement), and c) using different types of ants (forward ants and backward ants) to perform different tasks.

So far, we saw how discrete combinatorial optimization problem is solved using ACO framework. Ge *et al.* [15] described an ACO algorithm to Œnd global optima of a multi-extreme continuous space function and is described next.

2.3 Continuous Search Space

We now describe an ACO framework to Œnd global optima of a multi-extreme continuous space function $f(x)$. Here x is a n-dimensional vector. The Powell's method [23] is a numerical optimization technique to Œnd global optima of continuous multi-extreme functions. A drawback of this technique is that this method often gets trapped in local optima or by constraints in a region far from the optimal solution. Ge *et al.* [15] describe a hybrid ACO algorithm, which incorporates the Powell's method to solve this problem.

The Œrst step in their search involves dividing the solution space into several small regions R_i, i= 1, 2, ... and selecting an arbitrary point in each of these small regions. Their search algorithm comprises two basic steps: a) *global search* that uses ACO to identify one or more promising region $\in R_i$ for search and b) *local search* using the Powell method to perform optimization in a particular small region. Let m be the total number of ants and let $node_k(i)$ represent the new location of ant k that is currently located at node i. Here location of an ant means value of $f(x)$ for some vector x. First, they compute f_{avg} which represents the average of all function values of m ants. All ants whose current location is less than f_{avg} continues to perform global search where as those ants whose location value is greater than f_{avg} are perturbed probabilistically using the pheromone trail strength. The pheromone trail strength, denoted by τ, is updated using the following two equations:

$$\Delta\tau_k = c_1 f(x_i) \qquad (5)$$

$$\tau(j+1) = \rho\tau(j) + \sum_{k=1}^{m} \Delta\tau_k \qquad (6)$$

where c_1 is a positive constant, and $\rho \in (0, 1]$ is the pheromone trail decay coefŒcient. Thus, when $f(x_i)$ is large, then $\Delta\tau_k$ is also proportionally large, thereby, providing greater amount of pheromone to the optimal function value. Thus, the key idea presented is to use a *divide-and-conquer* approach to solve this problem using both global search (ants) and local search (Powell's Method).

2.4 Avoiding Local Optima in ACO Framework

We now describe a few advanced techniques that can be used to guide ACO to avoid local minima. Stutzle and Hoos [33] introduced the MAX-MIN ant system (MMAS) that improves the performance of the ACO by: a) stronger exploitation of the best solutions found during the search and b) search space analysis. The key to achieve best performance of ACO algorithm is to combine an improved exploitation of the best solutions found during the search with an effective mechanism for avoiding early search stagnation. The key differences between MMAS and a simple ACO are:
1. To exploit the best solutions found during an iteration or during the run of the algorithm, after each iteration only one single ant (*global-best* ant) adds pheromone.
2. To avoid stagnation of the search, the range of possible pheromone trails on each solution component is limited to an interval $[\tau_{min}, \tau_{max}]$.
3. The pheromone trails are initialized to τ_{max} to achieve a higher exploration of solutions at the start of the algorithm.
4. *Pheromone trail smoothing* (PTS) facilitates the exploration by increasing the probability of selecting solution components with low pheromone trail.

Ouyang *et al.* [28] proposed a new ACO framework called *multi-group* ant colony system to avoid stagnating state. Once a group of ants arrive at local optimum point, a new group of ants are initialized and the two groups together perform the search. If two groups are still in local optimum point state, a third group is initiated and three groups search. The algorithm iterates until global best optimum point is searched out.

In this section, we described the basics of ACO and several ideas to formulate different optimization problems within the ACO framework. In the next section, we describe our novel ACO framework for Boolean satisŒability.

3 SAT for Digital VLSI Circuits Using ACO

Most SAT problems in the electronic design automation (EDA) industry originates from digital VLSI circuits [35]. A traditional approach is to convert the circuit information to conjuctive normal form (CNF). However, doing so results in the loss of circuit structural information. With CNF form, it is not possible to take advantage of that information. Hence, it is always desirable to develop a circuit-based SAT solver [25, 35].

In this paper, our goal is to simultaneously solve multiple SAT instances for digital VLSI circuits using a novel ACO framework. We solve multiple Boolean SAT problems simultaneously because such a tool can then be easily extended for solving test generation problem. Boolean SAT instances are created using the ISCAS'89 benchmark circuits. For each primary output (PO)/pseudo-primary output (PPO) we obtain two Boolean SAT instances, B_{out} and $\overline{B_{out}}$ using the logic cone driving that particular PO/PPO, labeled out. Here B_{out} represents the Boolean *true* function represented by the signal line out and $\overline{B_{out}}$ represents the Boolean $false$ function represented by the signal line out. This means that for each circuit we will have a total of $2 \times (N_{po} + N_{ff})$ SAT instances, where N_{po} represents the number of POs and N_{ff} represents the number of Œip-Œop in the circuit. P_i^k will be used to denote the kth SAT problem, where i(=0, 1) represents the goal to generate Boolean i at the PO/PPO corresponding to the problem. We will now describe the key ingredients of our ACO framework: a) the

Authorized licensed use limited to: IEEE Xplore. Downloaded on November 21, 2008 at 11:41 from IEEE Xplore. Restrictions apply.

artiﬁcial pheromone trail, b) the *goal oriented ants*, and c) *pheromone trail updation*.

3.1 Artificial Pheromone Trail

We use two types of pheromone called 0-pheromone and 1-pheromone. For each signal line l in the circuit, we use two variables denoted by $p_0(l)$ and $p_1(l)$ to represent the pheromone trail strengths for 0-pheromone and 1-pheromone. The value $p_0(l)$ measures the degree of easiness for generating a Boolean 0 at signal line l and $p_1(l)$ quantiﬁes the degree of easiness for generating a Boolean 1 at signal line l. Pheromone strengths $p_0(l)$ and $p_1(l)$ will guide the ants to take the right path to obtain the input assignment.

We initialize the pheromone strengths as follows. First, we compute the SCOAP controllabilities $(C0(l), C1(l))$ [16] values for each signal line l in the circuit. We then compute $(C0MAX,\ C1MAX)$ where $C0MAX$ represents the maximum value of 0-controllability of all signal lines and $C1MAX$ represents the maximum of 1-controllability of all signal lines in the circuit. Then initial pheromone strengths, $p_0(l)$ and $p_1(l)$ for each signal line l is computed as

$$p_0(l) := 2 \times C0MAX - C0(l) \tag{7}$$
$$p_1(l) := 2 \times C1MAX - C1(l) \tag{8}$$

Our experiments indicate (see Section 4) that initializing the pheromone strengths using the SCOAP measures represents a good starting point for conducting the search.

3.2 Goal Oriented Artificial Ants

We use goal oriented ants i.e. each ant in our ant colony has a particular goal at a given point of time, which can be either to generate a Boolean 0 or a Boolean 1 at the current signal line. Each time when an ant moves from the output signal line of one gate to one of the input signal line (rules for ant movement is described in Section 3.3) of this gate, its goal changes according to the rules given in Table 3. For AND, OR, and BUF the output goal is same as that of the goal at the inputs. This is because for AND/OR/BUF to generate output 1(0), the input should be set to 1(0). But for NAND, NOR, and NOT generating 1(0) requires input(s) be set to 0(1). Currently, our ACO framework does not support gates such as XOR/XNOR, multiplexors, AOI gates etc. For designs containing these complex gates, one should use standard cells describing these complex gates using primary gates and ﬂatten the design.

Table 3: Rules to Describe Ant Goal

AND/OR/BUF		NAND/NOR/NOT	
Current Goal	New Goal	Current Goal	New Goal
1	1	1	0
0	0	0	1

3.3 Generating Ants and Ant Movement

For each SAT problem P_i^k, we generate an ant A_k whose goal is i. During each iteration, only a subset of these ants participate in the search because two ants starting from the same PO/PPO but with different goals are never generated. We do this because both ants will never be able to simultaneously succeed as it is not possible to generate both 0 and a 1 at a particular PO/PPO simultaneously. Thus, we only consider one ant for each PO/PPO. If an ant succeeds in achieving its desired goal, then we add a new ant starting from the same PO/PPO but with different goal. Thus, in the ﬁrst iteration we will have N_{INIT} ants where $N_{INIT} < N_{po} + N_{ff}$. After each iteration, since many ants will succeed in achieving their goal, new ants will replace the successful ants. If there are no more ants to replace, then in the subsequent iterations the number of ants will be lesser than N_{INIT}.

All ants start traversal from its initial location i.e. a PO/PPO in a concurrent fashion. The concurrent movement of ants are simulated using the pseudo-code in Figure 3. Ants move from one signal line to another based on the following movement rules. If the goal of the current ant is to generate a 1(0) at the output of AND(OR) gate or NOR(NAND) gate G, then the ant has to visit each one of the inputs of G. In that case, its new goal is given by the rules described in Section 3.2. However, if the goal is to generate a 0(1) at the output of AND(OR) gate or NOR(NAND) gate G, then the ant need to visit only one of the input of G with new goal determined again from the rules in Section 3.2. The ant will pick that input in which it is most easy to achieve the desired goal, which can be determined from the pheromone strength. Section 3.4 describes how pheromones are dynamically adjusted so that it will always give a reasonably good measure of the degree of difﬁculty of generating Boolean 1 and 0. When an ant A_k visits signal line l, we also register its current goal in a variable $RV(l)$. If multiple ants visit the same signal line l with different goals, then we register both the Boolean values by assigning the value 2 to $RV(l)$. These registered values $RV(l)$ for each signal line l is later used for updating the pheromone strengths.

```
//take a subset of ant and insert them into the queue
insertADifferentSubsetOfAntsInQueue( &q, A);
while(!isEmptyQueue(q))
{
    //obtain the next ant from the queue to simulate
    ant = getNextAntFromQueue(q);
    if( thisAntAchievedDesiredGoal(ant) )
        markThisAntDead(ant);
    else {
        simulateAntMovement(ant);
        insertTheAnt( &q, ant);
    }
}
```

Figure 3: Concurrent Ant Movement Using a Queue

3.4 Updating Pheromone and Generating Input Assignment

In this subsection, we will ﬁrst describe how to ﬁnd if a particular ant has achieved its desired objective. When a particular ant reaches a PI/PPI p, its goal is recorded in a variable $G(p)$. If multiple ants arrive at input p with different goals, then we count the number of times goal 1 was recorded labeled as $N_1(p)$ and number of times a goal 0 was recorded for the input p denoted by $N_0(p)$. If $N_1(p) > N_0(p)$, then $G(p)$ is assigned a Boolean value 1, otherwise we assign a Boolean value 0. When all ants reach the PI/PPI, then most of the bits in the vector G are speciﬁed by 1 or 0. All unspeciﬁed inputs are randomly assigned values 0 or 1. Vector

Authorized licensed use limited to: IEEE Xplore. Downloaded on November 21, 2008 at 11:41 from IEEE Xplore. Restrictions apply.

G constitutes one test vector. Thus, we see that the test vector automatically emerges out due to the ant movement. We then use a 3-valued logic simulator to Œnd out if a Boolean 1 or 0 is generated at each PO/PPO. From these values, we can Œgure out if a particular ant achieved its goal or not. If the desired goal is achieved then the ant dies. Otherwise, it remains in the subsequent iteration until it achieves its desired goal. To update the pheromone strength, we use variable $RV(l)$ in each signal line l (see Section 3.3). The 3-valued logic simulation of the test vector G will give us the logic value $V(l)$ generated at signal line l. Table 4 describes how $p_0(l)$ and $p_1(l)$ is updated for each signal line l, based on the values of $RV(l)$ and $V(l)$. In Row 1 of Table 4, $V(l)$=0, $RV(l)$=0 means that the goal at signal line l was 0 and the input assignment generated Boolean 0 at l. Since, this is a favorable outcome, we strengthen the 0-pheromone by incrementing the $p_0(l)$ value, where as $p_1(l)$ remains the same. In Row 2, $V(l)$=0, $RV(l)$=1 means that the goal was to generate Boolean 1 at l but the input assignments generated Boolean 0. Hence ants with the goal of Boolean 1 are penalized from moving towards l by decrementing $p_1(l)$, where as ants with goal of Boolean 0 are favored by incrementing $p_0(l)$. The other three rows in Table 4 can be similarly explained. In the last row, $Not\ Reg.$ means no Boolean goal was registered at signal line l. Thus, the Boolean value generated by the input assignment at signal line l is compared to the Boolean goal registered at l. This comparison is used to adjust the pheromone strengths that will correctly guide the ants in the subsequent iterations. The pseudo-code for our overall ACO algorithm is given in Figure 4.

Table 4: Rules to Update Pheromone Strength

$V(l)$	$RV(l)$	$p_0(l)$	$p_1(l)$
0	0	Increment by 1	Do Nothing
0	1	Increment by 1	Decrement by 1
1	0	Decrement by 1	Increment by 1
1	1	Do Nothing	Increment by 1
0/1	$Not\ Reg.$	Do Nothing	Do Nothing

```
AcoForSat (ckt)
1. Obtain logic cone driving each PO/PPO
2. Initialize pheromone strengths using eqn. 7 & 8
     and generate ants                      // see Sections 3.1 & 3.2
3. for i ← 1 to 100
4.      concurrentAntMovement();            // see Figure 3
5.      Obtain input assignment             // see Section 3.4
6.      Logic simulate                      // see Section 3.4
7.      Update pheromone                    // see Table 4
```

Figure 4: Pseudo Code for our ACO Framework

4 Results and Analysis

The proposed ACO framework was implemented in C programming language and ISCAS'89 benchmark circuits were used to validate the theory presented in this paper. We performed two experiments. In experiment 1, we initialized the pheromone values to a constant value and in experiment 2, pheromones were initialized using the SCOAP measures as described in Section 3.1. Table 5 shows the results of our experiments. Column labeled $Total\ SAT$ shows the total number of SAT instances generated. Columns under heading $Without\ SCOAP$ corresponds to the results of experiment 1 and $With\ SCOAP$ corresponds to the results of experiment 2. Columns labeled $Det.$ represents the total number of SAT instances proven satisŒable and columns labeled $Time$ represents the total CPU time in seconds. Columns under the heading $Spectral$ corresponds to the results of spectral ATPG, a state-of-the-art test generation tool for digital VLSI circuits that was run in combinational mode. We use spectral to compare our results because it also implements a non-deterministic algorithm using spectral transform and selŒsh gene meta-heuristics.

Our experiments indicate that SCOAP measures gives a good starting point for the ACO framework. Given the same amount of time, ACO framework with SCOAP measure is able to solve more number of SAT instances than the one without using SCOAP measure. We also see that our ACO framework consistently outperforms the spectral and selŒsh gene meta-heuristic. For the circut $s15850$, spectral ATPG tool had initialization problems and we are investigating its cause. Spectral technique requires initializing and building large matrices. This consumes very large CPU time. Since ACO framework does not need any compute intensive or memory intensive opteration, so it is signiŒcantly faster. One may argue that the proposed ant traversal is similar to a backtrace procedure of ATPG algorithm, but note that several ants concurrently traversing the netlist makes them different and more effective than any conventional ATPG search algorithm.

Disadvantages of our scheme are: a) it is an incomplete algorithm and cannot solve all SAT instances, and b) it is a non-deterministic algorithm and cannot prove that a SAT instance is unsatisŒable. Hence, a deterministic SAT Solver [26, 27] may outperform the ACO algorithm as it can prove that a SAT instance is satisŒable or unsatisŒable. However, note that our algorithm consumes extremely short CPU time. Thus, our algorithm can be used as a Œrst pass for determining satisŒability of large number of SAT instances. A second pass using a deterministic tool can be used to prove satisŒability of the instances aborted in the Œrst pass.

5 Conclusion and Future Work

In this paper, we presented an ant colony optimization (ACO) framework to simultaneously solve multiple Boolean SAT instances. Results of our experiments on scanned versions of ISCAS'89 benchmark circuits indicate that ACO algorithm can solve large number of SAT instances using very short CPU time. Applications of the proposed ACO algorithm are: a) Test generation, b) Generating patterns to detect multiple faults, and c) Compaction, all of which consumes extremely large CPU time. The proposed algorithm can be used as a Œrst pass to detect large number of faults using very short CPU time. A deterministic algorithm can be used to determine undetectable and hard to detect faults.

We also presented the basics of ACO and how different types of optimization problems can be formulated within the ACO framework. There are large numbers of optimization problem (see Table 2) in the Œeld of VLSI testing and ACO can be used for solving all of these problems. However, many researchers in this Œeld are unaware of the ACO framework because ACO was never applied in the Œeld of testing. Hence, we believe that this paper can open the gates for researchers solving different optimization problems that exists in VLSI testing.

References

[1] S. Alupoaei and S. Katkoori. Ant Colony Optimization Technique for Macrocell Overlap Removal. In *Proc. of the Int'l. Conf. on VLSI*

Authorized licensed use limited to: IEEE Xplore. Downloaded on November 21, 2008 at 11:41 from IEEE Xplore. Restrictions apply.

Table 5: Results of our ACO Framework (* see Section 4)

Benchmark	*Total SAT*	*Without SCOAP*		*With SCOAP*		*Spectral* [36]	
		Det.	*Time(s)*	*Det.*	*Time(s)*	*Det.*	*Time(s)*
s1494	50	42	0.11	43	0.19	43	1.01
s5378	456	362	1.14	370	0.86	365	23.92
s9234	500	430	2.21	458	2.33	455	57.09
s13207	1580	1261	4.02	1284	3.84	1280	400.01
s15850*	1368	1227	5.19	1245	5.76	0	0
s38417	3484	3165	13.81	3186	13.93	2400	2627.04
s38584	3460	2863	13.31	3084	13.11	2800	2729.24

Design, Jan. 2004.

[2] S. Barbagallo, G. Borgonovo, D. Grassi, D. Medina, F. Corno, P. Prinetto, and M. Sonza Reorda. Scan Chain Partitioning and Reordering Based on Layout Information: An Industrial Experience. In *Proc. of the Design Automation and Test in Europe Conf.*, pages 123–127, 1998.

[3] K. Chakrabarty. Design of System-on-a-Chip Test Access Architectures Under Place-and-Route and Power Constraints. In *Proc. of the Int'l. Test Conf. (ITC)*, pages 432–437, 2000.

[4] S. T. Chakradhar and V. D. Agrawal. A Transitive Closure Algorithm for Test Generation. In *Proc. of the IEEE/ACM Design Automation Conf.*, pages 353–358, June 1991.

[5] S. T. Chakradhar, V. D. Agrawal, and S. G. Rothweiler. A Transitive Closure Algorithm for Test Generation. *IEEE Trans. on Computer Aided Design*, 12(7):1015–1028, July 1993.

[6] S. T. Chakradhar, M. A. Iyer, and V. D. Agrawal. Energy Minimization Based Delay Testing. In *Proc. of the Int'l. Conf. on Computer-Aided Design*, pages 280–284, 1992.

[7] S. Devanathan and M. L. Bushnell. Sequential Spectral ATPG Using Wavelet Transform and Compaction. In *Proc. of the Int'l. Conf. on VLSI Design*, pages 407 – 412, 2006.

[8] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Dipartimento di Elettronica, Politecnico de Milano, IT, 1992.

[9] M. Dorigo, D. Caro, and L. M. Gambardella. Ant Algorithm for Discrete Optimization. *ArtiŒial Life*, 5(2):137–172, 1999.

[10] M. Dorigo, V. Maniezzo, and A. Clorni. Positive Feedback as a Search Strategy. Technical report, Dipartimento do Elettronica, Politecnico di Milano, IT, 1991.

[11] M. Dorigo and T. Stutzle. *Ant Colony Optimization*. The MIT Press, Cambridge, Massachusetts, 2004.

[12] F. Ducatelle, G. D. Caro, and L. M. Gambardella. Ant Agents for Hybrid Multipath Routing in Mobile Ad Hoc Networks. In *Proc. of the Conf. on Wireless On-demand Network Systems*, 2005.

[13] L. M. Gambardella, E. D. Taillard, and G. Agazzi. *New Methods in Optimization*. McGraw-Hill, 1999.

[14] L. M. Gambardella, E. D. Taillard, and M. Dorigo. Ant Colonies for the QAP. *The Operational Research Society*, 50(2):167–176, 1999.

[15] Y. Ge, Q. Meng, C Yan, and J. Xu. A Hybrid Ant Colony Algorithm for Global Optimization of Continuous Multi-extreme Functions. In *Proc. of the Int'l. Conf. on Machine Learning and Cybernatics*, Aug. 2004.

[16] L. H. Goldstein and E. L. Thigpen. SCOAP: Sandia Controllability / Observability Analysis Program. In *Proc. of the IEEE/ACM Design Automation Conf.*, pages 190–196, June 1980.

[17] P. P. Grasse. La reconstruction du nid et les interactions inter-individuelles chez les bellicostermes natalenis et cubitermes sp. la theorie de la stigmergie: essai d'interpretation des termites constructuers. *Insectes Sociaux*, 6(1):41–83, 1959.

[18] M. Gunes, U. Sorges, and I. Bouazizi. ARA - The Ant Colony Based Routing Algorithm for MANETS. In *Proc. of the ICPP Int'l. Workshop on Ad Hoc Networks*, 2002.

[19] M.S Hsiao, E.M Rudnick, and J.H. Patel. Application of Genetically Engineered Finite-state-machine Sequences to Sequential Circuit ATPG. *IEEE Trans. on Computer Aided Design*, 17(3):239–254, 1998.

[20] M.S. Hsiao, G.S. Saund, E.M. Rudnick, and J.H. Patel. Partial Scan Selection Based on Dynamic Reachability and Observability Information. In *Proc. of the Int'l. Conf. on VLSI Design*, pages 4–7, 1998.

[21] O. Khan and M. L. Bushnell. Spectral Analysis for Statistical Response Compaction During Built-In Self-Testing. In *Proc. of the Int'l. Test Conf. (ITC)*, pages 67–76, 2004.

[22] W. Li, S. Wang, S. T. Chakradhar, and S. M. Reddy. Distance Restricted Scan Chain Reordering to Enhance Delay Fault Coverage. In *Proc. of the Int'l. Conf. on VLSI Design*, pages 471–478, 2005.

[23] Bao lin Chen. *Theory and Algorithms of Optimization*. Tsinghua University, Beijing, China, 1998.

[24] P. Meksangsouy and N Chaiyaratana. DNA Fragment Assembly Using an Ant Colony System Algorithm. *The 2003 Congress on Evolutionary Computation*, 3(8-12):1756 – 1763, 2003.

[25] M.K.Ganai, L.Zhang, P.Ashar, A.Gupta, and S.Malik. Combining strengths of circuit-based and CNF-based algorithms for a high-performance SAT solver. In *Proc. of the IEEE/ACM Design Automation Conf.*, June 2002.

[26] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an EfŒcient SAT Solver. In *Proc. of the IEEE/ACM Design Automation Conf.*, pages 530–535, June 2001.

[27] J. P. M.Silva and K. A. Sakallah. GRASP: A Search Algorithm for Propositional SatisŒability. *IEEE Trans. on Computers*, 48(5):506–521, May 1999.

[28] J. Ouyang and G. Yan. A Multi-Group Ant Colony System Algorithm for TSP. In *Proc. of the Int'l. Conf. on Machine Learning and Cybernetics*, 2004.

[29] Zahra sadat Ebadi and Andre Ivanov. Design of an Optimal Test Access Architecture Using a Genetic Algorithm. In *Proc. of the Asian Test Symp.*, page 205210, 2001.

[30] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based Load Balancing in Telecommunication Networks. *Adaptive Behavior*, 5(2):169–207, Nov. 1996.

[31] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-like Agents for Load Balancing in Telecommunication Networks. In *Proc. of the Int'l. Conf. on Autonomous Agents*, 1997.

[32] J.P.M. Silva. Integer Programming Models for Optimization Problems in Test Generation. In *Proc. of the IEEE/ACM Asia PaciŒ Design Automation Conf.*, pages 481 – 487, 1998.

[33] T. Stutzle and H. Hoos. MAX-MIN Ant System and Local Search for the Traveling Salesman Problem. In *Proc. of the Int'l. Conf. on Evolutionary Computation*, 1997.

[34] G. Theraulaz and E. Bonabeau. A Brief History of Stigmergy. *ArtiŒ cial Life*, 5(2):97–116, Nov. 1999.

[35] F. L. Wang, K.T. Cheng, and R.C.-Y.Huang. A Circuit SAT Solver with Signal Correlation Guided Learning. In *Proc. of the Design Automation and Test in Europe Conf.*, pages 892–897, 2003.

[36] J. Zhang, M. L. Bushnell, and V. D. Agrawal. On Random Pattern Generation with the SelŒsh Gene Algorithm for Testing Digital Sequential Circuits. In *Proc. of the Int'l. Test Conf. (ITC)*, pages 617–626, Oct. 2004.

[37] W. Zou, S. M. Reddy, I. Pomeranz, and Y. Huang. SOC Test Scheduling Using Simulated Annealing. In *Proc. of the VLSI Test Symposium*, pages 325–330, May 2003.

Authorized licensed use limited to: IEEE Xplore. Downloaded on November 21, 2008 at 11:41 from IEEE Xplore. Restrictions apply.