Test Suite Skimming on Agent Based Model Using Maximum Clique on a Modified Bee Colony Algorithm

Dr. Vivekanandan and G. Keerthi Lakshmi

***Abstract*—** *The need for test case skimming is inevitable in almost all arenas of software testing owing to the business constraints on the drop dead date of the date of delivery to pass the QA cycle. Trade-offs are often needed to between time and the test coverage to ensure the maximum test cases are covered within the stipulated time, especially on systems that are just entering the production environment after getting promoted from the staging phase.*

The most important test cases are often not deemed to qualify under the sanity test suite and any bugs that creeped in them would go undetected until it is found out by the actual user at firsthand. Hence there arises a need to refine the test cases to accommodate the maximum test coverage it makes within the stipulated period of time.

An attempt is made in this paper to layout a testing framework to address the process of the test case skimming by following the maximum clique identification on an improved bee colony algorithm, where the optimum input to the test-bed is determined by selecting the test-cases that constitute the input.

This input heavily depends on determining the location of the result hive and sub-partitioning, thus test-cases with higher quality are picked up by the improved bee colony algorithm and provides better test efficiency. The Average Percentage of Conditions Covered (APCC) metrics have been used to show the effectiveness of proposed algorithm.

***Keywords*---** *Test case Skimming, Agent based Modeling, Software Testing, Bee Colony, Graph Maximization, Clique*

I. Introduction

To increase the reliability of any software application, stakeholders need automated and cost-effective test strategies that adequately test the continuously evolving application. One effective approach to testing of these applications is by adopting the agent based testing strategy, which reduces a large, unmanageable set of test-case inputs to a much smaller set that is likely to reveal bugs in the system under test.

Agile testing is a software testing practice that follows the principles of agile software development. Agile testing does not emphasize testing procedures and focuses on ongoing testing against newly developed code until quality software from an end customer's perspective results. Agile testing is built upon the philosophy that testers need to adapt to rapid deployment cycles and changes in testing patterns. Software is developed in incremental, rapid cycles. This results in small, incremental releases, with each release building on previous functionality. Each release is thoroughly tested, which ensures that all issues are addressed in the next iteration.

Testing out the hourly agile builds in a real-time system and taking out the best build that is deemed to get qualified as a stable build is a tedious and time bound task, where the heuristics are to be applied at tandem, which constitutes the most effective test cases that could pave way to catch the high priority bugs**.**

Simulated Bee Colony (SBC) algorithms model the behavior of honey bees and can be used to find solutions to difficult or impossible combinatorial problems. SBC algorithms are often called meta-heuristics because they provide a general framework and set of guidelines for creating a problem solution rather than providing a highly detailed solution prescription.

An attempt has been made in this paper to reduce the test suites on agile test process by embedding the maximum clique identification on a bee colony algorithm after modifying it to fit into the agile test process.

II. Literature Review

Population based algorithms can be classified by the nature of phenomenon simulated by the algorithm into two groups: evolutionary algorithms (EA) and swarm intelligence based algorithms [1], [2].

The most popular among EA is genetic algorithm (GA). GA attempts to simulate the phenomenon of natural evolution. A branch of nature inspired algorithms which are called swarm intelligence is focused on collective behavior of some self-organized systems in order to develop some meta-heuristics which can mimic such system's problem solution

Dr. Vivekanandan, School of Management, Bharathiar University, Coimbatore, E-mail: vivekbsmed@gmail.com

G. Keerthi Lakshmi, Research Scholar, Department of Computer Science,Bharathiar University, Coimbatore, E-mail: keerthi.gopal@gmail.com

abilities [3], [4].

Interaction between individuals locally with one another and with their environment contributes to the collective intelligence of the social colonies [5]. Even though there is no centralized component that controls the behavior of individuals, local interactions between all individuals often lead to the emergence of global behavior.

These characteristics of swarms inspired huge number of researchers to implement such behavior in computer software for optimization problems [6]. Flocking of birds and schooling of fish are examples of swarm systems. The classical example of a swarm is bees swarming around their hive but the metaphor can easily be extended to other systems with a similar architecture such as ants [7].

Nature inspired algorithms that are based on the social behavior of certain animals and insects can solve many complex problems such as the traveling salesman problem (TSP), vehicle routing, scheduling, networks design and many more [2].

Generally, such algorithms are applied to problems classified as NP-hard or NP-complete. Many practical problems in industry and business are in the class of intractable combinatorial (discrete) or numerical (continuous or mixed) optimization problems. Many traditional methods were developed for solving continuous optimization problems, while on the other hand, discrete problems are being solved using heuristics [8].

III. The Current System Under Test

The existing system where improvisation is sought is a pure collation of child web-agents which are geographically separated and are cloud enabled by providing Service Access Point (SAP) of the interfaces of the system.

The master-agent is placed at a location central to all these web agents, which trigger each of these child agents asynchronously. The web agents convey the results to the invoked agent each time when the tests are run.

IV. The BEE Colony & its Algorithm

Common honey bees such as *Apis mellifera* assume different roles within their colony over time. A typical hive may have 5,000 to 20,000 individual bees. Mature bees (20 to 40 days old) usually become foragers. Foraging bees typically occupy one of three roles: active foragers, scout foragers and inactive foragers.

A. Active Bees

Active foraging bees travel to a food source, examine neighbor food sources, gather food and return to the hive.

B. Scout Bees

Scout bees investigate the area surrounding the hive, often a region of up to 50 square miles, looking for attractive new food sources. Roughly 10 percent of foraging bees in a hive are employed as scouts.

C. InActive Bees

At any given time some of the foraging bees are inactive. These inactive foragers wait near the hive entrance.

When active foragers and scouts return to the hive, depending on the quality of the food source they've just visited, they may perform a waggle dance to the waiting inactive bees. There's strong evidence that this waggle dance conveys information to the inactive bees about the location and quality of the food source. Inactive foragers receive this food source information from the waggle dance and may become active foragers.

In general, an active foraging bee continues gathering food from a particular food source until that food source is exhausted, at which time the bee becomes an inactive forager.

The original Bee Colony Algorithm is as given below:

1. Initialize the population of solutions
2. Evaluate the population
3. Produce new solutions for the employed bees
4. Apply the greedy selection process
5. Calculate the probability values
6. Produce the new solutions for the onlookers
7. Apply the greedy selection process
8. Determine the abandoned solution for the scout, and replace it with a new randomly produced solution
9. Memorize the best solution achieved so far

V. Identifying the Modification Parameters

The original BCO algorithm is derived from nature. The algorithm can be directly applied to our Agent model testing process as it is still a numeric optimization problem, however the following traits could be considered for re-modeling:

A. The bee-hive construction is de-centralized and could be made "location-aware". Multiple hives are constructed at the places so that the time to travel and return from hive is minimized. In real world, bees could cover a distance of 36 km/hour, and their spermatheca size is much more such that the nectar collected could last for weeks if individually consumed.

B. Whereas, in the agent system, the test investigation real time data would span for a few megabytes and wiring the entire data over the network would result in a considerate amount of bandwidth and time loss.

C. A location could be made void if the nectar accumulation rate is dropping down than the below Global index minus the threshold rate and the newer areas could be explored at.

D. The stopping criteria is when the time for the testing has elapsed or when the entire coverage is achieved or when the nectar accumulation rate continuously falls less than threshold in a row.

E. The probability of success of communication becomes no matter in this system as opposed to the real system where "Waggle Dance" from the scout carries a ratio of miss.

VI. The Modified Algorithm

1. Evaluate the population – identify the master bee for the synchronization
2. Engage the captain scout bees to determine the organized colony sets. The colony sets must match the thresholds specified as a part of configuration.
3. Scout performs waggle dance until the required number of bees needed starts following the nuclear team.
4. The branch hives are constructed and the greedy selection process is applied
5. The sum of the paths are quantified to reach the target

   Aj1+ Aj2 + Aj3 …..<= N

6. If Target is not met, then proceed to the step 4, else publish the target path to the captain bee with the result of the analysis.,

Once the algorithm is complete, it constructs N independent colonies which are located at discrete location from each other. The next step is to construct a graph out of the colonies and identify the maximum clique of it, which denotes the best path that is to be adopted.

In computing the clique, a greedy algorithm is to be arrived at, which starts with a simple, incomplete solution to a difficult problem and then iteratively looks for the best way to improve the solution. The process is repeated until some stopping condition is reached. The greedy heuristic successively chooses a vertex of the maximum degree in the sub-graph composed of the vertices joined with all vertices already included into the formed clique.

Let G(V; E) be a simple undirected graph, V = {C1,C2,C3,C4 ….Cn}. The adjacency matrix of G is a matrix $A_G = (a_{ij})_{mxn}$, where $a_{ij} = 1$ if (i,j) € E, and aij = 0 if (i; j) 2= E. The set of vertices adjacent to a vertex i € V will be denoted by N(i) = {J € V : (i; j) € E } and called the neighborhood of the vertex i. A clique Q is a subset of V such that any two vertices of Q are adjacent. The maximum clique problem asks for a clique of the maximum cardinality. This cardinality is called the clique number of the graph and denoted by !(G).

The complete algorithm for this final step is as below:

Input: a graph G(V; E).
Output: a maximal clique Q.

1. Construct the vector of vertex degrees d € R^n such that:

$d_i = |N(i)|$.

2. Set $V_1 := V$; k := 1; Q := Ω.
3. Choose a vertex v_k € v_k such that d_{vk} is greatest.
4. Set Q := Q U $\{U_k\}$.
5. Set $V_{k+1} := V_k$ λ $N(V_k)$.
6. 6. For each j € V_{k+1},

dj := dj-|(V_k € V_{k+1}) λ N(j)|.

7. If V_{k+1}!= Ω, then k := k + 1 and go to 3.
8. STOP.

VII. The Implementation

The proposed BCO algorithm has been implemented in C-sharp compiler comprising of about 3000 LOC. Work of incorporating agent into the code is in progress so that this code will be able to judge the input automatically and hence it could be used to solve test suites of larger size and it will then require minimum human interface.

The time of transmission of data is currently calculated and is applied to the agent data structure, however in the real time it would be computed by the intelligent agents later in the code.

To check the effectiveness of the algorithm, an agent system that was in use for online theatrical bookings was attempted at the affiliates web sites by hosting the sites over a cloud and exposing the location tracker to find out the nearest request.

Step 1 - Construct Autonomous Bee Colonies

Out of 20 requests that arrived over a span of 98 minutes at the peak center of the week, the location tracker with the load balancer identified the following data resources:

Request #	Route Direction
1,3,7,9,14,15,18	North
2,5,10,20	South
6, 17	East
4,5,8,11,12,13,16,19	West

Thus, 5 colonies were formed with the 100 simulated bees with the merge and join of the four directional colonies owing to the failure to meet the threshold guidelines and subsequent route directions were aimed at any of these colonies where the requests came from.

The scout agent was initiated to perform the request analysis and would trigger the active agents whose slots are empty to service the request, once the requests are serviced, the scout agent would pass a signal to the master. Based on

the code coverage obtained, the colonies are dissertained as follows:

Colony #	Approx time of completion (seconds)	Path opted
C1	121	P1, P3 P9 P15 P 18 P14 P7
C2	36	P2 P5 P20
C3	45	P5 P8 P10
C4	82	P16 P13 P12 P11
C5	41	P6 P17 P10

Step 2 - Translate Colony Nodes to Graphs

The next step is to discretely transform each of the colonies as graph nodes and establish a better reachability amongst them, which would be the skimmed test suite.

Once the individual colonies are complete then the minimal path of graphs is obtained by the weighted edges, as below:

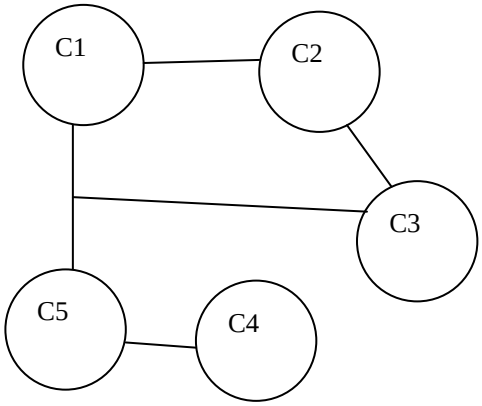

Figure 1: The output of the Bee Colony Algorithm Plotted as a Undirected Graph whose Maximum Clique is to be Computed and thus Provide the Optimal Skimmed Test Case

The connectivity between the nodes C3-C4 is disregarded as it loses on the threshold with a weight of 6 (distance between P10- P16, provided the bees picked up the path linearly, not randomly).

The aim here is to find out the maximum clique of the graph which provides the maximum atomicity of the test cases in the suite. A clique is a subset of a graph where every node is connected to every other node.

Thus the following is the best choosen clique of C1 → C2→C3 and the test completes best at 121+36 +45 = 202 seconds as against providing 38% improvement by using APCC metrics.

VIII. Conclusion

The test case skimming method using maximum clique on a Modified Bee Colony algorithm is a powerful and emerging way of refining the test suite and is often mis-interpreted as inaccurate way of testing by practitioners, however both travel in different directions and target varied audience.

The availability of tools and techniques to achieve this technique shall further pool in more resources into this arena where lot of improvisations to the current design is possible.

IX. Further Improvements

The greedy maximum clique algorithm explained here can be modified to produce better results by adding a tabu feature. Tabu algorithms maintain a list of recently used data and possibly a list of recently seen solutions. Data in the tabu list isn't used to construct new solutions. Additionally, pleateau search can be used when the algorithm is unable to improve its current solution, which produces a new solution without going backward, such as dropping a node in the maximum clique problem.

References

[1] Xin-She Yang, Nature-Inspired Metaheuristic lgorithms, Luniver Press, 2008.
[2] Johann Dréo, Patrick Siarry, Alain Pétrowski and Eric Taillard, Metaheuristics for Hard Optimization, Springer Berlin Heidelberg, pp. 1-19, 2006.
[3] Muddassar Farooq, Bee-Inspired Protocol Engineering: From Nature to Networks, Springer, 2008.
[4] Xin-She Yang, Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms, Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, Volume 3562/2005, No. 10.1007/b137296, 2005, pp. 317-323.
[5] Saif Mahmood Saab , Nidhal Kamel Taha El-Omari, Hussein H. Owaied, Developing optimization algorithm using artificial bee colony system, UbiCC Journal - Volume 4, No 5, 2009, pp. 391-396.
[6] Tricia Rambharose and Alexander Nikov, Computational intelligencebased personalization of interactive web systems, WSEAS Transactions on Information Science and Applications, Vol. 7, Issue 4, Apr 2010, pp. 484-497.
[7] N. Buniyamin, N. Sariff, W. A. J. Wan Ngah, Z. Mohamad, Robot Global Path Planning Overview and a Variation of Ant Colony System Algorithm, International Journal of Mathematics and Computers in Simulation, Vol. 5, Issue 1, 2011, pp. 9-16.
[8] T. Y. Chen,Y. L. Cheng: Global optimization using hybrid approach, WSEAS Transactions on Mathematics, Vol.7 ,2008 , pp. 254-262.