CHAPTER 2

LITERATURE SURVEY

2.1 APPLICATION OF EXISTING APPROACHES IN SOFTWARE TEST OPTIMIZATION

2.1.1 Conventional Methods

The conventional methods for testing the software component are “static testing by humans”. Under this, desk checking, code walkthrough, formal inspection and some of the brute-force techniques are applied.

The manual testing process is slow, costly and is not effective in optimal test case generation. Exhaustive testing, that is testing infinitely is also not possible. There is no proof of completion of testing and only a less maintenance of record for detected errors. One of the most important problems in manual testing is to identify the ways of reducing manual workload, cost and time (Gerlich 2007).

Fewster and Dorothy (1999) discussed in their book, the effectiveness of applying automated testing tools to reduce the problems involved in manual testing. However, many of the automated testing tools available in the market lack scalability, intelligent decision making ability and general applicability.
Recently, various techniques have been applied for software test data selection, reduction and prioritization. A number of empirical studies comparing these techniques in terms of their defect-detection and coverage based effectiveness have been reported by many researchers.

2.1.2 Algorithmic Based Approaches

Benson (1981) performed an experiment in which assertions were used in conjunction with search techniques in order to test the Software under Test (SUT) automatically. In this experiment, errors were introduced in the program randomly using an error generation method. Then assertions were written for the SUT and tested using two methods. In the first one, they divided the range of input variables and selected test cases from within the sub-ranges. In the second approach, a search algorithm from optimization theory is used. This approach used assertions to define an error function and then maximizing its value. The results indicated that, the search technique coupled with assertions can identify more number of errors in the SUT.

Richard et al (1993) proposed a constraint based testing approach to detect specific types of programming faults. The conditions were encoded as mathematical systems of constraints in terms of program symbols. They applied a set of tools called Godzilla, to automatically generate constraint systems and solve them to create test cases for use by the Mothra testing system. They identified the effectiveness of the test cases in terms of mutation adequacy.

Agrawal (1994) used the notion of dominators and super blocks to derive coverage implications among the basic blocks with the goal of reducing coverage requirements for testing a program.
David et al (1998) submitted a technical report on applying statistical methods in software testing. Their report addressed the alternatives for exhaustive software testing based on statistical methods, including multivariable analysis, design of experiments, coverage designs, usage models, and optimization techniques. Their approach provided quantitative measures of quality and reliability, as well as statistical measures and confidence levels if the given SUT implements its functional specification correctly. The overall goal of their approach is to ensure software quality and to develop methods for software conformance testing based on relevant statistical techniques.

Martina and Antonia (2003) exploited entity subsumption and spanning trees to determine the reduced set of coverage entities such that coverage of reduced set implies the coverage of unreduced set.

Beydeda and Gruhn (2003) proposed a binary search based test case generation approach. The primary use of the BINTEST algorithm is in testing the methods of a class. The proposed algorithm is classified as dynamic path oriented approach. The authors have addressed the problem of dynamic test case generation approaches and provided a solution with binary search based test case generation algorithm. The path to be covered is considered step by step and test cases are then searched to fulfill them. In their paper, the proposed binary search algorithm which is used to determine the test cases, requires certain assumptions but allows efficient test case generation.

Sreedevi et al (2004) applied greedy approach for web based testing. They treated each web session as an object, URLs used in sessions as attributes. Then they select one test case from next-to-bottom concept and removed the redundant test cases.
Prowell (2005) focused on the usage of Markov chain usage models and concurrency operators in Test case Generation Language (TGL). The test cases were generated from the Markov model and then they have been represented by means of TGL in XML based notation. He introduced several operators in the XML code to represent the test cases. In his previous paper (2004), he indicated a stopping criterion for statistical testing.

Ostrand et al (2005) applied a negative binomial regression model to predict faults in the SUT before the next release of the Software to the customer side. They calculated the predictions based on the code in the current release, faults and the modification history from previous releases. The proposed approach was also applied to a much smaller input set that contained fault data from integration testing.

Tallam and Gupta (2005) of the University of Arizona, in their paper analyzed the application of Greedy algorithm in selecting efficient test data from an infinite number of possible ones. They applied classical Greedy Heuristic for Set Cover algorithm to reduce the test suite size. Their approach, named as DelGreedy, selected the optimal test cases based on the number of requirements covered by each test case. According to their approach, they pick test case $t_i$ that covers most requirements. Then the covered requirements were deleted from the set of requirements. The algorithm is again applied until all requirements are covered.

Greg Dennis et al (2006) described an approach for checking the methods of a class against a full specification. It shares with traditional model checking - the idea of exhausting the entire space of executions within some finite bounds. Then, with traditional verification idea of modular analysis, a method is analyzed, in isolation, for all possible calling contexts. The analysis involves an automatic two-phase reduction: first, to an intermediate form in
relational logic, and second, to a Boolean formula which is then handed to an off-the-shelf SAT solver.

Ingo et al (2007) presented the generation of test cases for model based code generators. The approach heavily makes use of the fact that both input and output of the code generation are executable. It enforces the automatic generation of test cases to cover the high variability of models.

Xiao (2007) analyzed the evaluation of optimization algorithms when used in goal oriented automated test data generation. They identified that goal-oriented approach is a promising approach to devise automated test data generators. They conducted a comparative study about the effectiveness of the commonly used optimization techniques.

Li and Sun (2007) proposed an algorithm based on reversed binary tree for automated test data generation. They claimed that their algorithm can automatically find out all of the feasible paths in the program from the source node to the base node and can automatically generate test data for each founded feasible path.

Xiao Ma et al (2007) proposed a methodology based on prioritized constraints with data sampling scores to generate test data even when a set of constraints is not solvable.

Wes Masri (2007) empirically evaluated several test case filtering techniques based on coverage and profile distribution based techniques. Also, they compared the performance of random sampling against filtering techniques in exercising various program elements. The observations indicated that, distribution based filtering techniques did not perform well when compared to coverage based filtering techniques.
Andrea et al (2008) focused on the difficulties involved in testing, container classes with nature inspired algorithms. They applied input space reductions and a novel testability transformation to aid the search algorithms.

Ramon and Jose (2008) described two approaches which employ Estimation of Distribution Algorithm as a Meta-Heuristic technique for test data generation. The approach first extracts a region in the initial search space that incorporates static information from the SUT. If this method is not enough, then a grid search method is applied.

2.1.3 Program Slicing Based Approaches

Elaine and Bingchiang (1991) applied partition testing strategies, which divide a program's input domain into subsets with the tester selecting one or more elements from each sub domain, for test data reduction. They analyzed the conditions that affect the efficiency of partition testing and compared the fault detection capabilities of partition testing and random testing.

Whitaker and Thomason (1994) described a method for statistical testing based on a Markov Chain model of software usage. They discussed the significance of the Markov Chain in two ways. They first generated test input sequences from multiple probability distributions. Then, the test input sequences generated from the first chain, are applied to the software. The test sequences were represented by means of stochastic model and were used to create a second Markov chain to encapsulate the history of the test, including any observed failure information. The influence of the failures was assessed through analytical computations on this chain. They also derived a stopping criterion for the testing process based on a comparison of the sequence generating properties of the two chains.
Forgács and Bertolino (1997) proposed an approach for feasible test path selection using principle slicing. The focus was based on deriving feasible test paths from the software by selecting those paths which reach a specified program point with a number of influencing predicates that is as low as possible. Principle slicing supplies a program slice to a specified program instruction with an almost minimum number of influencing predicates. The approach statically derived program slices with a near minimum number of influencing predicates using both control flow and data flow information.

Hierons et al (1999) proposed a slicing based approach for mutation testing. In their approach, they employed slicing to identify equivalent mutants which according to them will not be identified by the normal test cases.

Chandrasekhar et al (2002) presented a novel framework called ‘Korat’ for automated testing of Java programs. Given a formal specification for a method, ‘Korat’ used the method precondition to automatically generate all (non-isomorphic) test cases up to a given small size. ‘Korat’, then executed the method on each test case, and used the method post condition as a test oracle to check the correctness of each output. To generate test cases for a method, Korat constructed a Java predicate (i.e., a method that returns a boolean) from the method’s precondition.

Zhenqiang et al (2003) described a test coverage analysis based on Program Slicing. They applied Slicing as a way for test case optimization. They decomposed a large system into a series of smaller ones by applying program slicing technique. Then they computed the statement, branch and path coverage of the generated subsystems. Finally the coverage of the original system was obtained by the total coverage of the subsystems, according to their importance. In their approach, the entire work is depending on program slicing.
Willem et al (2004) described the test input generation using Java Path Finder (JPF). They applied model checking and symbolic execution to generate test inputs in order to achieve coverage of code that manipulates complex data structures. Their proposed framework used method preconditions to initialize fields with valid values and method post conditions as test oracles to test a method’s correctness. The testing criterion was encoded as a set of properties the model should check for. Their framework applied lazy initialization and symbolic execution.

David et al (2005) tested the feasibility of applying Bounded Exhaustive Testing (BET) to test the systems having structurally complex inputs. They used tools like ‘TestEra’ and ‘Alloy’ to get the inputs. They tested the proposed approach to the Galileo dynamic fault tree solver. Their paper focused primarily on the input generation bottleneck. Their work was based on the idea of factoring specifications in a principled way to reduce the load on a constraint solver while enabling the complete input space to be reconstituted by a simple postprocessor. They suggested that this capability could provide a general-purpose optimization technique for constraint-based test input generation.

2.1.4 Knowledge Based Approaches

Deason et al (1991) proposed a rule-based test data generation approach as an alternative approach to path/predicate analysis or random test data generation. They constructed a rule-based test data generator to generate test cases using rules. And then they compared the effectiveness of the said approach against randomly generated test cases. They confirmed by means of basic coverage metrics; rule based test data generation is better than random test data generation.
Anderson et al (1995) investigated the application of Neural Networks in test case generation. Their work concentrated on experimental effectiveness analysis and prediction. They used a neural network as a classifier to learn about the system under test and to predict the fault exposure capability of newly generated test cases. They described attributes of test cases to the neural network as inputs and relate them to resulting faults (neural network outputs). Inputs are test case length and various coverage metrics relevant to the testing strategy used. Outputs are levels of severity of faults detected. They range from severe (level 1) to “correct operation” (level 4). Then, they trained the network to recognize relationships between test case descriptors and faults.

Pargas et al (1999) have applied Genetic Algorithm for test data generation. Their paper presented a goal oriented technique for automatic test data generation that uses genetic algorithm which is guided by the control dependencies in the program to search for test data to satisfy test requirements. In their approach, GA conducts its search by constructing new test data from previously generated test data that were evaluated as good candidates.

Memon et al (2002) proposed an AI based planning approach for GUI based testing. In their approach, the actions were modeled at different levels of abstraction. The operators were modeled at level n and one or more methods were modeled at level ‘n-1’. Their test case generation technique enabled efficient application of planning by first creating a hierarchical model of a GUI based on its structure. The GUI model consists of hierarchical planning operators representing the possible events in the GUI. The test designer defined the preconditions and effects of the hierarchical operators, which are input into a plan-generation system. The test designer also creates scenarios that represent typical initial and goal states for a GUI user. The
planner then generates plans representing sequences of GUI interactions that a user might employ to reach the goal state from the initial state.

Saraph et al (2003), in their paper presented a novel methodology to identify important test cases automatically using Neural Networks (NN). In their approach, they reduced the number of test cases by identifying input-output relationships. A ranked list of features and equivalence classes for input attributes of a given code are the main outcomes of their methodology. The approach was based on construction, training, pruning and rule-extraction of a neural network. The approach employed two feature ranking methods based on sorting and pruning. Also, a rule extraction method has been employed for expressing the I-O relationships. But, it needs a lot of work to convert the hidden unit activation values from continuous to discrete. Also, it itself needs a clustering algorithm and a rule-extraction algorithm and only then, the test cases can be generated.

Diaz et al (2003), in their paper, explained the application of Tabu search in software test data generation. Their objective was to obtain branch coverage using program control graph. The Tabu search consists of two lists for memorizing the good and worst tests respectively. The goal of the approach is to minimize the size of the test suite.

Anders et al (2003) proposed a time-optimal test case generation using UPPAL. Their paper demonstrated how it is possible to generate time-optimal test cases and test suites, i.e. test cases and suites that are guaranteed to take the least possible time to execute. The required behavior is specified using a deterministic and output urgent class of UPPAAL style timed automata. The UPPAAL model checking tool implements a set of efficient data-structures and algorithms for symbolic reach ability analysis of timed automata. The application of fastest diagnostic trace facility of the UPPAAL tool is used to generate time optimal test sequences. Test cases can either be
selected through manually formulated test purposes or automatically from three natural coverage criteria such as transition or location coverage of the timed automata model.

Li and Lam (2004) reported on the application of ACO as a supplementary optimization stage for finding sequences of transitional statements in generating test data for evolutionary testing. They used the pheromone value of each ant to identify the visited and unvisited nodes. They consider the problem of simultaneously dispatching a group of ants to cooperatively search a directed graph ‘G’. The ants in their paradigm can sense the pheromone traces at the current vertex and its directly connected neighboring vertices, and leave pheromone traces over the vertices. Each ant keeps its own private sets while the public set is left on the graph for all of the ants to share. Ants can sense the pheromone levels on the graph, and modify the public set in the exploration of the graph.

Baudry et al (2005) discussed the application of bacteriologic algorithm in software test data generation and selection. Mutation analysis is the basic concept behind it. This approach is used to build confidence on test cases, normally used to qualify the unit test cases for object oriented classes and gives the programmer the useful feedback on the fault revealing power of their test cases. It also offers an estimate of how many new test cases they need, to better test a given software component. The bacteriologic algorithm takes as input an initial set of test cases and output a good set of test cases. This algorithm evolves incrementally. That is the algorithm builds final set incrementally by memorizing test cases that can improve the set’s quality. Stopping criteria like after ‘x’ number of generations, when the solution set reaches a solution set reaches a minimum fitness value, if the set’s fitness value has not changed for a number of generations and so on.
In a technical report of Abhas and Pankaj Jalote (2005), they proposed a dynamic test case generation using Neural Networks (NN). According to them, the execution of the blocks in a program is controlled by various branch predicates. The execution of a particular block depends on the evaluation of the corresponding branch predicate guarding the block. They defined a function for each of the branch predicate. The program is instrumented so as to record the function evaluation for the corresponding inputs. A large record of such mappings from external inputs to the evaluation of branch functions can then be modeled using a neural network. They trained the network in order to approximate the function and predict the correct the results. In this process, they used random input values and test them on the function models.

Last et al (2006) have introduced a new, computationally intelligent approach to generation of effective test cases based on a novel, Fuzzy-Based Age Extension of Genetic Algorithms (FAexGA). In their approach, the crossover probability $P_c$ is determined by a Fuzzy Logic Controller (FLC). The FLC state variables include the age and the lifetime of the chromosomes to be selected for crossover (parents) along with the average lifetime of the current population. They identified good test cases from the bad test cases based on their fault revealing capability. The approach is application dependent. The test configurations must be modified manually to construct the application specific fuzzy rules. Hence fuzzy rule base is not a generalized one and it is an application dependent one. The fuzzification and defuzzification processes consumes a lot of time. One cannot generalize the results for all the applications.

Du (2006) applied a machine learning based approach to identify the value of test cases. In his paper, he proposed a framework for value based
software test data generation. His proposed framework has principles that greatly help to improve return on investment by means of value of test cases.

Pravin et al (2006) described the value of test cases in software testing. They listed up the components of test cases, the quality factors of test cases and test cases format. They also explained a way of choosing a test type based on its preference value. Also, they analyzed the different ways of improving test cases using language features, controlling length of the test cases, cumulative cases based on pros and cons, applying templates and clones to improve the productivity of test cases.

Shradhanand et al (2007) investigated the application of fuzzy logic in software development. In their approach, they specified test selection as the activity of choosing a finite set of elements (e.g., requirements, functions, paths, data) to be tested out of a typically infinite number of elements. Test selection is often based on an adequacy or coverage criterion that is met by the elements selected for testing. The fact that only a finite subset of elements is selected inevitably introduces a degree of uncertainty regarding whether all defects in the system can be detected. One can therefore associate a probability value with a testing criterion that represents one's belief in its ability to detect defects. Their approach dealt with modeling uncertainty using fuzzy logic.

Manish et al (2007) evaluated the application of AI planning techniques in automating the testing process. They proposed a framework for an Automated Planning System (APS) to test each and every module of the SUT. They applied a MEA-Graph plan algorithm to automatically generate test data which is nothing but the sequence of steps or actions to transform the current state of the system to some desired goal state. They avoided the state-space explosion problem by regressing goals over actions during graph expansion phase of planning.
Doungsa-ard et al (2007) proposed a method to automatically generate test data from gray-box testing. They applied GA to generate the test cases from specification. UML state diagram is used as a specification for generating test data.

Hélène et al (2007) investigated the implementation of Simulated Annealing to test data generation. They applied the concept of landscape to the testing problem. They introduced a new measure Generation Rate of Better Solutions (GRBS) to monitor the convergence of the search process and implementation of stopping criteria.

Harman et al (2007) identified that there has been no investigation on the relationship between the size of the input domain and performance of search based algorithms. They applied static analysis to remove irrelevant variables during test data generation process thereby reducing the size of the search space. They analyzed the impact of local and global search algorithms in domain reduction for real world examples.

Anastasis and Andreas (2008) proposed a dynamic test data generation framework based on genetic algorithms. The proposed approach has a program analyzer and a test case generator. The program analyzer extracts statements and variables and isolates code paths and creates control flow graphs. The test case generator they proposed utilized two optimization algorithms namely Batch-Optimistic and Close-up to produce near to optimal set of test cases with respect to edge/condition coverage criterion.

Raquel et al (2009) applied a test reduction strategy based on Scatter search approach to select few efficient test cases based on a test adequacy criterion. In their paper, they presented and analyzed two versions of the approach for the automatic generation of software test cases using branch coverage based adequacy criterion.
2.2 APPLICATION OF PROPOSED SEARCH APPROACHES IN OTHER NP-HARD PROBLEMS

2.2.1 Application of Intelligent Agents (IA)

Jeremy and Azim (2002) demonstrated a collaboration protocol based on a conditional measure of agent effectiveness for flow optimization of railroad traffic. Because agent effectiveness is not directly measurable, a suitable metric for agent effectiveness is introduced. Where typically agents run with uniform frequency, the collaboration protocol schedules the agents with a frequency proportional to their expected effectiveness.

Christopher (2003) applied testing and monitoring activities on intelligent agents. Since, the intelligent agents are applied to various domains; their correctness needs to be assured. This paper proposed a methodology to assure the correctness of the agents.

Dhavachelvan and Uma (2003) presented a Multi-Agent based approach to enhance the definition for class testing in object-oriented paradigm. The integrated framework has been built on two existing testing techniques namely Mutation Testing and Capability Testing. In both the cases, testing is carried out at Autonomous Unit Level (AUL) and Inter-Procedural Level (IPL). Mutation Based Testing-Agent and Capability Assessment Testing-Agent have been developed for performing AUL testing and Method Interaction Testing-Agent has been developed for performing IPL testing.

Kung (2004) applied intelligent, autonomous agents to Web applications testing which involve knowledge-driven, labor intensive activities. The proposed framework is based on the Belief-Desire-Intention (BDI) model of rational agents and the Unified Modeling Language (UML).
They described how web applications testing can be modeled and reasoned using the framework.

Rem et al (2004) described an UML based software engineering methodology for agent factory. They applied agent oriented software engineering (AOSE) methodology that employs a synthesis of the unified modeling language (UML) and Agent UML to support the development of multi-agent systems.

Mangina (2005) proposed an intelligent agent-based platform for implementation, where the approach of integrating the use of two or more techniques is taken, in order to combine their different strengths and overcome each other's weaknesses and generate hybrid solutions. The integration of various intelligent techniques is a very important way forward in the next generation of monitoring systems. Flexible software systems require a distributed architecture where each element works autonomously and co-operatively. According to their paper, this advanced architecture can be created by making use of Intelligent Agent technology, which will facilitate the creation of autonomous software components to utilize a wide variety of artificial intelligence techniques for different situations. This is in marked contrast to conventional centralized stand-alone applications.

Miao et al (2007) employed a methodology to design a formal open framework based on agent for automatically testing web applications. In their framework, each test task corresponds to a role and the agent takes this role to achieve its test task or cooperates by interaction with other agent to finish the test tasks. The agent can not only join or leave agent society at will, but also take or release roles at run time dynamically.
Zhiyong et al (2008) have enhanced the Prometheus Design Tool (PDT) which is an agent design tool, by adding a feature that allows the automated unit testing of agents that are built from within PDT.

Zhu et al (2008) analyzed the problems existing in the current development on the application of the agricultural expert system. They proposed the development for agriculture-specific software using agents. They discussed the application of agents, and constructed an agent-based Agricultural Expert System Inspection Tool is constructed. In addition, their paper addressed the outlook in application, potential problems and the development trend of multi-agent-based inspection software for the agricultural expert system.

### 2.2.2 Application of Hybrid Genetic Algorithm (HGA)

Walker et al (1994), proposed the use of Hybrid Genetic Algorithm to genetic sequencing problem. Map building is an example of a difficult sequencing problem which requires some form of search to find a good solution from a large problem space of feasible solutions. The authors described the development of a hybrid genetic algorithm (HGA) suitable for tackling the problem. The results of applying the HGA to a set of map data were presented.

David (1996) applied Hybrid Genetic Algorithm to Airline Crew Scheduling. His approach found the optimal solution for half the problems, and good solutions for nine others. The results were compared to those obtained with branch-and-cut and branch - and -bound algorithms.

José et al (2002), proposed a hybrid genetic algorithm for manufacturing cell formation. This paper presented a new approach for obtaining machine cells and product families. The approach combined a local...
search heuristic with a genetic algorithm. Computational experience with the algorithm on a set of group technology problems available in the literature was also presented. The approach produced solutions with a grouping efficacy that is at least as good as any results previously reported in literature and improved the grouping efficacy for 59% of the problems.

Kim et al (2003) applied a hybrid genetic algorithm and neural network approach in activity-based costing. This paper proposed hybrid artificial intelligence techniques to resolve these two problems. Genetic algorithms were used to identify optimal or near-optimal cost drivers. In addition, artificial neural networks were employed to allocate indirect costs with nonlinear behavior to the products.

Weijin et al (2004) described Hybrid genetic algorithm research and its application in problem optimization. In order to explore a new resolution, the author proposed a combining algorithm for structural optimization, which is based on genetic algorithm and gradient algorithm. Gradient algorithm was used to superpose, and the result got was used to improve the herd of the genetic algorithm. The superior genetic algorithm was compared with the root of the gradient algorithm and the best point was chosen to be the incipient point of the next step of the super position.

Pushkaryova (2004) applied Hybrid Genetic Algorithm in CAD applications. The designed set of algorithms and programs was intended for automated design of optimal trajectories of closed paths in a bi-variate area given by a technological map of cutting. The results obtained give a reason to believe that application of artificial intelligence technologies and parallel computing in CAD will allow to effectively solving the complicated problems of discrete continuous structure.
Yici (2005) applied HGA to the Crosstalk Aware Track Assignment Problem. This paper presented a genetic algorithm hybridized with a constructive procedure and reported its application on the crosstalk aware track assignment problem. In this algorithm, only dominating elements were encoded as chromosomes, on which genetic operators worked to explore the solution space, while other elements were determined using constructive method. With proper dominating elements identification, the proposed approach essentially searches a much smaller space without trivial operations, efficiently generating competitive solutions with an effective constructive procedure. Experimental results on a set of industrial instances and ISPD98 benchmarks showed that the proposed algorithm reduces both capacitive and inductive coupling in acceptable running time. It is probable that the proposed approach provided a practical way for the application of genetic algorithm on large scale engineering problems.

Arcuri and Yao (2007) applied memetic algorithm for generating test data in object oriented systems.

Ahmet et al (2007), applied HGA based approach for electrical sounding method. Their paper suggested some hybrid genetic algorithms, derived from evolution theories, to overcome this problem.

William et al (2008) applied hybrid genetic algorithm for multi-depot vehicle routing (VRT) problem. Their paper focused on the VRP with multiple depots, or multi-depot VRP (MDVRP). To deal with the problem efficiently, two hybrid genetic algorithms (HGAs) were developed in this paper. A computational study is carried out to compare the algorithms with different problem sizes. It is proved that the performance of HGA2 is superior to that of HGA1 in terms of the total delivery time.
Zhao et al (2009) applied HGA for the Traveling Salesman Problem with Pickup and Delivery. In the proposed algorithm, a novel pheromone-based crossover operator is advanced that utilizes both local and global information to construct offspring. In addition, a local search procedure is integrated into the GA to accelerate convergence. The proposed GA has been tested on benchmark instances, and the computational results show that it gives better convergence than existing heuristics.

2.2.3 Application of Artificial Bee Colony (ABC)

Artificial Bee Colony optimization that belongs to non-pheromone based swarm intelligence algorithms is considered suitable for many optimization based problems.

Dušan et al (2006), proposed a bees algorithm to solve difficult combinatorial optimization problems such as transportation problems. In their paper, in addition to proposing the Bee Colony Optimization (BCO) as a new metaheuristic, they also described two BCO algorithms called the bee system (BS) and the fuzzy bee system (FBS). In the case of FBS, the agents (artificial bees) use approximate reasoning and rules of fuzzy logic in their communication and acting. In this way, the FBS is capable to solve deterministic combinatorial problems, as well as combinatorial problems characterized by uncertainty.

Karaboga et al (2007) applied Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. In their paper, they showed superior performance of ABC in solving a set of constrained optimization problems. In the same year, they proposed the use of ABC in solving numerical function optimization. They compared the efficiency of ABC with other optimization algorithms such as GA, PCO and PS-EA and proved that ABC outperformed them.

Mohammad et al (2007) have applied honey-bee mating optimization algorithm on clustering in data mining. They applied honey-bee mating technique to avoid local optima in cluster analysis. They also compared the efficiency of the approach with other approaches such as GA, SA, TS, and ACO, on several well known data sets. Their finding showed that the proposed algorithm works well than the best one.

Wong et al (2008) presented an improved Bee Colony Optimization algorithm with Big Valley landscape exploitation as a biologically inspired approach to solve the Job Shop Scheduling problem. They compared the experimental results of the proposed algorithm with Shifting Bottleneck Heuristic, Tabu Search Algorithm and Bee Colony Algorithm with Neighborhood Search on Taillard JSSP benchmark and they showed that it is comparable to these approaches.


Alok Singh (2009) applied Artificial Bee Colony (ABC) Algorithm to the leaf-constrained minimum spanning tree problem. Given an undirected, connected, weighted graph, the leaf-constrained minimum spanning tree (LCMST) problem seeks on this graph a spanning tree of minimum weight among all the spanning trees of the graph that have at least ‘l’ leaves. In this paper, he proposed an artificial bee colony (ABC) algorithm for the LCMST problem. He demonstrated the superiority of the new ABC approach over all the other approaches based on computational results.

Karaboga (2009) proposed a new design method based on artificial bee colony algorithm for digital IIR filters.
2.3 SUMMARY

The summary of the literature survey is as follows:

- The software test optimization problem is much more difficult to solve and cause discontinuities in the objective function. And it is known to be Non-Polynomial (NP) hard.

- Brute-Force techniques are computational intensive procedures for solving complex and industrial strength software.

- Several algorithms have been proposed for specific types of problems but there is still a need for a more effective solution to more general problems.

- Research on software testing focuses mostly on test data generation and reduction and there is still a need for developing models for problems such as test sequence, test case and test suite optimizations.

- Several meta-heuristic search algorithms have been proposed to solve software test data generation.

- Knowledge based approaches have been extensively applied to software test data generation and optimization problems.

- There are some other meta-heuristic search approaches like intelligent agents; hybrid genetic algorithm and artificial bee colony have been emerged as potential techniques to solve many NP-hard problems.

- The survey on other meta-heuristic search techniques is an eye-opener to apply them for Software test optimization problem which is also NP-hard.