

A Wide Range Survey on Test Case Generation

M. S. Arunkumar

*Computer Science and Engineering
Surya Engineering College
Erode, India
arunkumarmsster@gmail.com*

Dr. R. Sasikala

*Information Technology
K.S.R College of Technology
Erode, India
sasikarathi@gmail.com*

Abstract

The survey paper categorizes, compares and summarizes articles related to Test Case generation methods and various Data mining algorithms. Testing is an important phase of software development, to maintain quality control and reliability of end products. As testing is an important phase in software development test case should be chosen effectively. Test cases illustrate tests that need to be run on the program to verify that program run as anticipated. The survey is done with the aim of integrating test case generation and data mining methods. Data mining method helps in obtaining reduced test suite which in turn facilitates mining and knowledge extraction from test cases. The integration and survey assist to select the best test case for testing process.

Index Terms: Data Mining, Test Case, Testing.

Introduction

Software organizations spend most of their money in testing related activities. Testing is an examination to provide stakeholders with information about product quality or service under test. Testing can also be stated as the process of verifying and validating software program or application, that meets its business and technical requirements, whether software works as expected. Testing is one of the most important phase of software development which helps in gaining consumer confidence. Hence it has to

be done carefully as software passes this phase then it means it is almost ready for delivery to customers. Hence, while choosing a test case for the testing purpose the developer should be very careful as it may distract the entire development process if it is chosen wrongly. Once test case is chosen carefully the testing process can be carried out at its level best. Sometimes the generated test case may be redundant which may cause the testing process to choose the same test cases repeatedly for the testing purpose. Software testing can be done in two ways. They are manual testing carried out by humans and automated testing by system or software. Manual testing requires a tester to play the role of an end user and use almost all features of the software application to ensure correct behavior. In Automated testing the software is tested to control the execution of tests.

A test case helps in verifying the entire coverage of the applications and tests all probable combinations in the application. It is an indication of the conclusion of the testing stage to a degree such that it reflects the areas under which the application would work better. It also helps the user to effortlessly reproduce the steps that were taken to uncover a fault that was detected during test. Different techniques for generating test cases have been proposed through the years. Generally test case is the description of the test. Test cases have several components to describe an input, event or action and anticipated response to check whether the application is working correctly or not. Test cases are stemmed from use cases and also from system requirements. One of the main advantage of generating test cases from requirements specification and design is that it will help the software or test engineer to uncover problems as early as possible. As the specifications are created early in the development cycle it will be ready for use before the programs are constructed. Generating test cases in advance also helps the testers to find ambiguities and inconsistencies in the requirements specification and design documents. Early detection of faults will definitely reduce the cost of building the software system.

Data mining refers to extracting or mining the knowledge from large amount of data. The term data mining is appropriately named as “Knowledge mining” or “Knowledge mining from data”. The huge volume of available data and the advanced storage technologies has made it promising for organizations to accumulate huge volume of data at lower cost. This stored data is utilized in order to extract useful and actionable information which is the goal of data mining process. Data mining can also be stated as the process of exploration and analysis by automatic or semiautomatic means of large quantities of data in order to discover meaningful patterns and rules.

Data mining is the subfield of computer science which involves pattern discovery from large data sets. This advanced analysis aims to extract information from large data set and convert it into an easily understandable structure for further use. The techniques used are at the seam of artificial intelligence, machine learning, statistics, database systems and business intelligence. At another angle, data mining can also be stated as essential process where intelligent methods are used in order to extract the patterns. The five major elements of data mining are,

1. Extract, transform, and load transaction data into data warehouse.
2. Various data are stored and managed using multidimensional database system.

3. Impart data access to business analysts and information technology professionals.
4. Examine the data by application software.
5. Present the data using graph or table hence it can be easily understand by all.

Data mining tasks can be classified in to two categories as descriptive and predictive. Descriptive mining tasks depict the most general property of data in database. Predictive mining tasks carry out inference in order to make predictions. The intention of a data mining task is normally to create either a descriptive or predictive model. A descriptive model presents data in a succinct way, the most important characteristics of the data set. It is basically an abstract of the data points, making it possible to study significant aspect of the data set. Conventionally, a descriptive model is initiated through data mining which is a bottom-up approach where the data speaks for itself. This undirected approach explores patterns in the data set but leaves the interpretation of the patterns to the data miner. The intention of a predictive model is to permit the data miner to predict an unknown value of a specific variable (target variable). If the target value is a real number then the task is regression or if it is a predefined number of discrete labels then the data mining task is classification.

Data mining is done to find patterns that are statistically reliable, unknown previously, and actionable from data. Data mining is an interdisciplinary subfield of computer science which involves computational process on large data sets. It aims in advanced analysis to extract information from a data set and transform it into an understandable information for further use. Different data mining technologies and functionalities are available which can be applied to choose most relevant and reduced set of cases for the purpose of testing. An efficient test case generation process followed by some of the data mining algorithms can be used to obtain reliable test case that result in a better approach.

Literature Survey

Test Case Generation

In this paper, survey is done on test case generation as a part of the research work.

A. Specification Based Test Case Generation

In [2,1] a novel technique has been proposed which implements predefined state based specification to generate test cases from UML state charts. UML test was carried out by integrating a test data generation tool with Rational Rose and a case study of Cruise Control System was discussed briefly. The testing process generated 34 non redundant test cases and the cruise control system has 7 functionalities, 174 blocks and 184 decisions. In order to test the system 24 faults were manually injected into the system. Experimental results demonstrate that full predicate test detected all faults, transition pair found 18 and statement coverage found 16. In this particular work, the author has

concentrated on UML statechart to get specifications. The coverage with test cases is measured using Atac and it covers 163 blocks i.e. 89% and 155 decisions i.e. 95%.

B.Path Oriented Test Case Generation

The algorithm proposed in [3,1] uses binary search strategy in path oriented test case generation. Main advantages of this method are it does not need any parameter calibration and does not require any optimizing techniques for test data generation. The algorithm uses dynamic path coverage technique for generating test cases. The issue of dynamic test case generation is addressed and a solution is provided with binary search based test case generation algorithm. The path that is to be covered is considered step by step and test cases are then generated to fulfill them. The binary search is used to determine test cases which require certain assumptions but test cases are generated efficiently.

C.Random Technique For Test Case Generation

Random technique in [5,4] makes use of one's assumption about the application to generate the test cases. The following are the classes to be tested and various test inputs are provided to assist in finding the faults. Auto test is the frame work used here for validation purpose. Auto test helps in predicting number of issues and number of undetected defects. The auto test classifies the test case as passed(no exception), unresolved(precondition violation in method under test), or failed(other exception). The create test method depicted below contains the main loop of testing strategy used in this technique and at each step it selects a method for testing and then executes this method.

```

Create_test(timeout):
  from
  initialize pool
  until timeout
  loop
  m:=choose(methods under test())
  create_test
  for method(m)
  end

```

D. Ant Colony Based Test Case Generation

In this approach [6] UML state chart diagram and ACO are used for generating test cases. The proposed approach has two main advantages of which the first one is direct usage of the UML artifacts created in the software design processes and the later one is automatically generated test sequence is always feasible, non redundant and achieves all stage coverage criterion.

The UML state chart diagram is converted to a direct graph which can be specified as $G=(V,E)$, Where V is set of vertices of the graph and E is a set of edges of the graph. A UML statechart diagram can be interpreted as a directed graph where the vertices are interpreted as the states and the edges are interpreted as the transition between the states. Consider the problem of dispatching a group of ants to search a

graph cooperatively. The ants in our paradigm can sense the pheromone traces at the current vertex and its directly connected neighboring vertices and leave pheromone traces over the vertices.

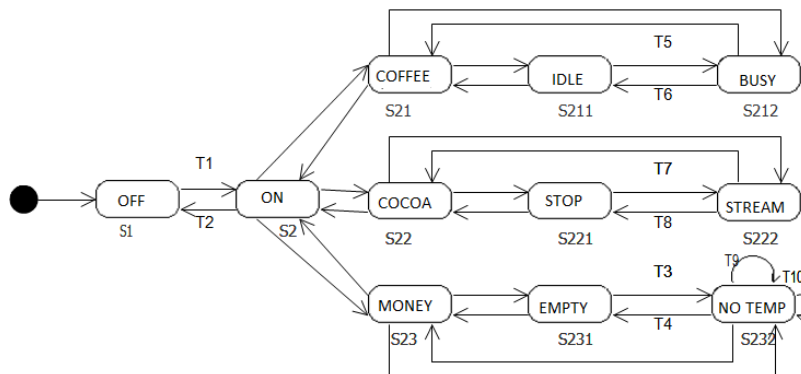


Figure 1: CCVM graph from UML State chart

An ant at a vertex α of the graph is associated with four tuples, Vertex Track Set S_k , Connection Set T_k , Target Set D_k , Pheromone Trace Set P . Each ant has its own private sets S_k , D_k and T_k and the public set P is shared by all ants on the graph. Ants will sense the pheromone level on the graph and update P in the exploration of the graph. The following algorithm explores the directed graph:

Algorithm for ant k

1. Evaluation at vertex α

Update the Track - Push the current vertex α into the track set S_k .

Evaluate Connections - Evaluate all connections to the current vertex α to determine T_k . The procedure involves evaluation of all possible transitions from the current states α to other neighboring states, using the state transition table associated with the UML Statechart diagram.

Sense the Trace - For the non-negative connections in T_k , the ant senses and gathers the corresponding pheromone levels P at the other ends of the connections.

2. Move to next vertex

Select Destination - The following prioritized rules are used in ant's selection:

- Select the vertex V_i with the lowest pheromone level $P(V_i)$ sensed from the current vertex α
- If vertices V_i and V_j shares the same lowest pheromone level $P(V_i) = P(V_j)$, but $T(V_i) = 0$ and $T(V_j) = 1$, select V_i
- If vertices V_i and V_j shares the same lowest pheromone level $P(V_i) = P(V_j)$ and $T(V_i) = T(V_j)$, randomly select one vertex

Destination β is the vertex selected using the above rules.

Update Pheromone - Update the pheromone level for the current vertex α to

$$P(\alpha) = \max(P(\alpha), P(\beta)+1) \text{ if } T(\beta) = 1$$

or

$$P(\alpha) = \max(P(\alpha), P(\beta)+1)+TP \text{ if } T(\beta) = 0$$

where TP is a high pheromone level which decays in one iteration of the two steps, namely, TP quickly decays to 0 before ant's next move at the end of Step 2

Move - Move to the destination vertex β , set $\alpha := \beta$, and return to Step 1.

In the algorithm the variable TP is used to avert the ant from doing redundant moves. Several ants can be used to explore the converted graphs simultaneously in order to speed up the exploration process and to produce shorter test sequences. When there is cooperative exploration at α an ant with a higher priority leaves and sets pheromone level for α first, followed by lower priority ants.

E. Model Based Test Case Generation

Model based test case generation is also one of the efficient test case generation techniques that help in deriving optimal set of test case. The Following Analysis examines various work evolved under model based test case generation.

Test Case Generation Based on Use case and Sequence Diagram [8,7] exemplify that test cases are derived from analysis artifacts such as use cases, sequence diagrams and the constraints specified across all these artifacts. Test Sequences is generated by initiating the derivation of Use case Dependency Graph (UDG) from use case diagram and Concurrent Control Flow Graph (CCFG) from the corresponding sequence diagrams. Focus the process of testing on sequences of messages among objects of use case scenarios. The testing strategy here uses full predicate coverage criteria for deriving test cases. Automating the entire process of test case generation is always an important issue. As manual testing is time consuming and error-prone, it is necessary to automate the testing process. Therefore with the continuous increase in software sizes the automatic test case generation is gaining importance. Test case generation technique proposed here can be used for integration and system testing accommodating the object message and condition information associated with the use case scenarios. Thus test cases generated are suitable for detecting synchronization and dependency of use cases and messages, object interaction and operational faults.

Automatic test case generation using Unified Modeling Language (UML) state diagrams [9,7] proposed an automated technique for model based test case generation. The approach known as the control and data flow logic available in the UML state diagram are used to generate test data. The state machine graph is traversed to select conditional predicates on every transition. And test cases are generated by applying function minimization technique on the transformed predicates. Test cases generated here satisfies transition path coverage criteria and can be used to test class as well as cluster-level state-dependent behaviors. The test data generated by combining this approach is verified based on path coverage. The various steps involved are, the first step is to select a predicate on a transition from a UML state machine diagram. In the next step, the selected predicate is transformed to a predicate function. In the third step, the test data generated corresponding to the transformed predicate function approach

can handle various change events, time events transition with guards and achieves transition path coverage which reduces the number of test cases that achieve transition path coverage by testing the borders that are determined by simple predicates.

Model-based software development focuses on establishing models of the system to be constructed. This approach helps to elaborate the most important properties of the software before proceeding with the implementation. The approach specified in Test Case Generation from UML State Machines [10,7] explains test cases not only include the stimuli to trigger the system under test, it also includes possible right observations for automatic evaluation of test case execution. Unlike classical Harel Statecharts, state machines behave asynchronously which is really a challenging issue for automatic test case generation. The TEAGER Tool Suite employs the automatic test case generation, execution and evaluation to prove the applicability of test approach [10,7]. In our approach UML state machines are used in the quality assurance process which serves as a specification for the anticipated reactive behavior of the system. Relevant and interesting inputs for a test case can be selected and the possible correct observations for given input can be calculated. They allow evaluating test executions automatically which is generally difficult and time consuming task. The approximation process here is more practical and it is possible to control this process depending on the time and computation power to invest.

Automated-Generating Test Case Using UML Statecharts Diagrams [11,7] experimented on the automatic testing technique to solve partially the testing process. This approach helps in automatic generation and selection of test cases from UML state chart diagrams. In the first step the UML state chart diagram is transformed into intermediate diagram called Testing Flow Graph (TFG). In the second step test case is generated from TFG using the testing criteria that is the coverage of the state and transition of the diagrams. In the final step the fault revealing power of our test cases is performed using mutation analysis. Specification based testing uses information derived from a specification in order to help testing and to develop the program. Testing activities includes designing test cases which are sequence of inputs, executing the program with test cases and verifying the results of this execution. Test cases are generated automatically from UML specification with the aid Rational Rose tool. The effectiveness of test cases are measured based on their fault detection capability. Simple test experiments carried out shows the high effectiveness of the generated test cases. However, wide ranging experiments are needed to have more confidence of the testing process and to measure its effectiveness.

F. Scenario Based Test Case Generation

A Novel Approach for Scenario-Based Test Case Generation [12,7] proposes a scenario based testing for generating test cases from test scenarios. UML activity diagrams describe the realization of the operation in design phase and also support description of parallel activities and synchronization aspects involved in different activities perfectly. Scenario testing works best for complex transactions or events, for studying end-to-end delivery of the benefits of the program, for exploring how the program will work in the hands of an experienced user, and for developing more persuasive variations of bugs found using other approaches. Methodology involved in

analyzing sequence and class diagram, Test unit definition, Search of setting and interaction categories, test case construction. Advantage of this methodology is that it handles the complicity of nested fork-join pair which is more often overlooked by other approaches. It overcomes the limitations of nested fork-join and loops.

Transformation based Approach to Generating Scenario-oriented Test Cases from UML Activity Diagrams for Concurrent Applications [13,7] employs UML Activity diagram for generating test case. A paper on “A Transformation-based Approach to Generating Scenario-oriented Test Cases from UML Activity Diagrams for Concurrent Applications” promotes transformation-based approach to generating scenario-oriented test cases for testing concurrent applications modeled by UML Activity Diagrams. Concurrent behavior is nondeterministic it’s testing is more difficult than the testing of common control flows or data flows. When activity diagrams are used to model concurrent applications under test, a challenge of generating test cases from activity diagrams relies in their non-structural properties. The approach first transforms a UML activity diagram specification into an intermediate representation via a set of transformation rules. From the intermediate representation author constructed a set of test scenarios with respect to the given concurrence coverage criteria. Finally, derive a set of test cases from the constructed test scenarios. The approach employs transformation to resolve the nonstructural problem with activity diagrams, and can generate test cases on demand to satisfy a given concurrence coverage criteria and hence the number of resulting test cases is controllable. Scenario oriented test cases from UML activity diagram specifications, with an emphasis on testing concurrent applications modeled using UML activity diagrams.

G. Genetic Based Test Case Generation

Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm [14,7] discussed is used to generate optimal test cases which also can be consider as data mining approach. The objects here exhibit dynamic behavior due to internal and external stimuli. And a method has been proposed for generating test cases by analyzing the dynamic behavior. The tree crossover one of the genetic algorithm technique is used to bring out all possible test cases from the given object diagram. Experimental results show that generated test cases have the capability to reveal 80% fault in the Unit level and 88% fault in the integration level.

We have viewed testing an application as traversing a path through the DFS (Depth First Search) for a binary tree to generate appropriate and adequate test cases. The mutation testing conducted has yielded 80.3% effectiveness in the actual testing process carried out with the generated test cases. Methodology is useful to generate test cases after the completion of the design phase and errors could be detected at an early stage in the software development life cycle.

H. Category Partition Test Case Generation

A Choice Relation Framework for Supporting Category-Partition Test Case Generation [15] proposes a method to capture the constraints among various values of the parameters called as choices. We express these constraints in terms of relations

among choices and combinations of choices called as test frames. Here uses theoretical backbone and techniques for consistency checks and automatic deductions of relations. Using algorithms test frames are created and the algorithm used has resource constraints specified by software testers, hence maintaining the effectiveness of the test frames generated. If the testers do manually then the constraints be ineffective and produce human errors. Hence, finally the constraints are captured among choices in a systematic manner which improves the effectiveness and efficiency of the complete test frame. Then enable the software tester to specify the relative priorities of the choices, there by subsequent formation of complete test frames.

Data Mining Methodologies

A. The K-Means Algorithm

The k-means algorithm is a simple iterative method to partition a given dataset into a user specified number of clusters, k . The algorithm operates on a set of d -dimensional vectors, $D = \{\mathbf{x}_i \mid i = 1, \dots, N\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i th data point. The algorithm is initialized by picking k points in \mathbb{R}^d as the initial k cluster representatives or “centroids”. Initial centroids are selected by random sampling from the dataset, setting them as the solution of clustering a small subset of the data or disquieting the global mean of the data k times. Then the algorithm iterates between two steps till convergence, the first step is called as Data Assignment in which each data point is assigned to its closest centroid, with ties broken subjectively. This results in partitioning of data. In the next step, known as relocation of means each cluster is relocated to centre of all data points assigned to it. If data points come with a probability measure then the relocation is to the expectations of the data partitions. The algorithm converges when the assignments no longer change. Note that each iteration needs $N \times k$ comparisons, which determines the time complexity of one iteration. The number of iterations required for convergence varies and may depend on N , but as a first cut, this algorithm can be considered linear in the dataset size. A detailed history of k means along with descriptions of several variations is given in [17,16].

B. Decision Trees

A decision tree is a graph like structure which includes chance event outcomes, resource costs, and utility and determines the course of action and probability. It is mainly used for classification where every node in decision tree denotes an attribute value, every branch denotes outcomes and tree leaves denotes class distribution. Decision trees separate the input space into cells where each cell belongs to each single class. The partitioning or separation be represented by sequence of tests. In business perspective decision tree can be viewed as creating a fragment of data from original data set. Each interior node in the decision tree tests variable and the branches are labeled with possible results on each test. The leaf nodes correspond to the cells and specify the class only if respective node is reached [19,18]. The classification is

done by starting from root node and proceeding based on the outcomes until a leaf node is reached. Pruning is the technique which is used in decision tree in order to reduce the size. Decision tree is a predictive model where each branch and leaves represent the separation of data set with the classification. In business perspective, decision tree can be viewed as creating a fragment of data from original data set. Hence marketing managers make predictions about their business from fragment of customers, products and sales region. These segments were derived from the decision tree. The main advantages are easy to understand even a hard problem and can be combined with other decision techniques.

C. Neural Network

Neural Network or an Artificial Neural Network(ANN) is enthused from animal's nervous system that detects patterns. The network is composed of elements working in parallel to solve a specific problem. The neural network breakthroughs in real world problems like customer response, prediction, fraud detection etc. Neural networks process information in a similar way as human brain. Data mining method such as artificial neural networks are able to model the relations that occur in data gathering and can be used for increasing intelligence in the business applications [20,18]. This predictive modeling method generates very intricate models that are very difficult to understand even by experts. Neural Networks are used in stock market, image compression etc. Artificial neural network is the signal processing technology which is a powerful tool for pattern recognition, decision problem or predication applications. ANN is fit, nonlinear system that learns to perform a function from data. The adaptive phase is normally training phase where system parameter will change during operations, parameter are fixed after the completion of training. If the problem is complex then using ANN model is accurate, the non linear characteristics of ANN provides the flexibility to achieve input output map Artificial Neural Networks provide user the capabilities to select the network topology, performance parameter, learning rule and stopping criteria.

D. Genetic Algorithm

Genetic Algorithm is used to get a better solution by combining one or more solutions like nature thus combining the DNA of living beings [21,18]. It is used as a problem solving technique but provided optimal solution. Since it is a general algorithm, it works in all search space and generates an exact solution. It uses the principles of selection and evolution to generate more solutions to a given problem. No mathematical analysis is needed. Crossover and mutation model combines the features of their parents and interrupts the original problem. Hence Genetic Algorithm is a best way of using this technique to solve the problem while having unknowns. Genetic Algorithm be used in State Assignment Problem, Economics, Scheduling, CAD, etc.

References

- [1] M.Prasanna, S.N.Sivanandam, R.Venkatesan, R.Sundarrajan, "A Survey on Automatic Test Case Generation", *Academic Open Internet Journal*, volume 15, 2005.
- [2] Jeff Offutt, Shaoying Liu, Aynur Abdurazik, Paul Ammann, March 2003, "Generating Test data from State based Specifications", *The Journal of Software Testing, Verification and Reliability*, 13(1):25-53.
- [3] Sami Baydeda, Volker Gruhn, 2003, "BINTEST – binary search-based test case generation", In *Computer Software and Applications Conference (COMPSAC)*, IEEE Computer Society Press, 2003.
- [4] Karambir, Kuldeep Kaur, "Survey of Software Test Case Generation Techniques", *International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 3, Issue 6, June 2013.
- [5] Ilinca Ciupa, Alexander Pretschner, Andreas Leitner, Manuel Oriol, Bertrand Meyer: "On the Predictability of Random Tests for Object-Oriented Software" *International Conference on Software Testing, Verification, and Validation*, 2008.
- [6] Huaizhong Li and C. Peng Lam, "Software Test Data Generation using Ant Colony Optimization", *World Academy of Science, Engineering and Technology, International Journal of Computer Information Systems and Control Engineering*, Vol:1 No:1, 2007.
- [7] A.V.K.Shanthi, D.Parthiban, G.MohanKumar, "A Survey of UML-Based automatic Test Cases Generation for Software Testing", *International Proceedings of Computer Science and Information Technology*, Vol.No.41, June 2012.
- [8] Santosh Kumar Swain, Durga Prasad Mohapatra, and Rajib Mall, "Test Case Generation Based on Use case and Sequence Diagram", *Int.J. of Software Engineering, IJSE Vol.3 No.2*, July 2010.
- [9] P. Samuel, R. Mall, A.K. Bothra, "Automatic test case generation using Unified Modeling Language (UML) state diagrams ",*Published in IET Software*.
- [10] Dirk Seifert, "Test Case Generation from UML State Machines", inria-00268864, version 2 – 23, Apr 2008.
- [11] Supaporn Kansomkeat and Sanchai Rivepiboon, "Automated-Generating Test Case Using UML Statechart Diagrams ", *SAICSIT*, 2003.
- [12] Baikuntha Narayan Biswal, Pragyan Nanda, Durga Prasad Mohapatra, 2008 IEEE, "A Novel Approach for Scenario-Based Test Case Generation", *International Conference on Information Technology*.
- [13] Chang-ai Sun, 2008 IEEE, "Transformation-based Approach to Generating Scenario-oriented Test Cases from UML Activity Diagrams for Concurrent Applications", *Annual IEEE International Computer Software and Applications Conference*.

- [14] M. Prasanna and K.R. Chandran, "Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm ",*Int. J. Advance. Soft Comput. Appl.*, Vol. 1, No. 1, July 2009.
- [15] T.Y.Chen, Pak-Lok Poon, "A Choice Relation Framework for Supporting Category-Partition Test Case Generation", *Ieee Transactions on Software Engineering*, Vol.29, No.7, July 2003.
- [16] XindongWu , Vipin Kumar , J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg, "Top 10 Algorithms in Data Mining", Springer, December 2007.
- [17] Jain AK, Dubes RC (1988) "Algorithms for clustering data". Prentice-Hall, Englewood Cliffs.
- [18] Nikita Jain, Vishal Srivastava, "Data Mining Techniques: A Survey Paper", *International Journal of Research in Engineering and Technology*, Vol-2, Nov 2013.
- [19] Lior Rokach and Oded Maimon,"Data Mining with Decision Trees: Theory and Applications(Series in Machine Perception and Artificial Intelligence)", ISBN: 981-2771-719, World Scientific Publishing Company, 2008.
- [20] R. Andrews, J. Diederich, A. B. Tickle, "A survey and critique of techniques for extracting rules from trained artificial neural networks", *Knowledge-Based Systems*,vol.- 8,no.-6, pp.-378-389,1995.
- [21] AnkitaAgarwal, "Secret Key Encryption algorithm using genetic algorithm", vol.-2, no.-4, ISSN: 2277 128X, *IJARCSSE*, pp. 57-61, April 2012.