Bio Inspired Approach for Generating Test data from User Stories

A.Tamizharasi[a], Dr. P.Ezhumalai[b], S. Remya Rose[c], P.Suresh[d], S. Logesswari[e]

[A]
Department of Computer Science and Engineering,RMD Engineering
College,Chennai, India.
[b]Department of Computer Science and Engineering,RMD Engineering College,Chennai, India.
[C]Department of Computer Science and Engineering,RMD Engineering
College,Chennai, India.
[d]Department of Computer Science and Engineering, KPR Institute of Engineering and Technology, Coimbatore, India
[e]Department of Computer Science and Engineering,RMD Engineering College,Chennai, India.

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: In Agile model where the software prototypes are developed frequently and also rapidly, testing becomes more critical. Generating an effective Test case for complex system is a challenging task involved in software testing. The major research challenge in this area includes the test case generation with limited resources, identifying the essential functional requirement that plays a crucial role and automation of the test case generation process. To solve this issue, a hybridized bio inspired approach is proposed to generate test cases from the user stories which accepts the business requirements as input, processed using NLP and develop functional test cases from it. The proposed algorithm is compared with other existing algorithms and the experimental results proved that the proposed algorithm is more efficient in many cases.

Keywords: Software Testing, Agile model, Test case generation, Bio inspired approach, NLP.

1. Introduction

Software Testing is the one of the most important phase in software development which consumes nearly 40-60% of effort, time and cost. Due to the end users urge to complete the project in short span with high quality and defects free, the testing activity has to be started as early as possible to fix the bugs at early stage. Generating an effective Test case plays a major role in software testing.

Test generation process deals with creation of a set of testing conditions which can be used for validating the adequacy of the application. Different techniques like model based technique which generates the test cases from the UML models, search based test generation which uses meta heuristic techniques that direct the search towards the potential areas of input space, random approaches that generates test cases based on assumptions, Goal based test data generation approach that cover a particular section, statement or function, specification based techniques that generates test data based on the formal requirement specifications, has been used but still there are a lot of research issues available.

Various metaheuristic techniques that have been applied in test case generation includes Genetic Algorithm, Particle Swarm optimization,cuckoo search, Ant colony optimization etc. guided by the fitness function for determining the quality of the search results. Recently, the meta-heuristic algorithms are used for generating test cases for multiple path coverage in one run.

Many previous researches have been conducted for generating the test cases from UML diagrams say Activity diagram, use case and state chart diagrams etc[2]. UML diagrams reduce the problem's complexity with increase in the product sizes and complexities. However, the approaches used previously did not provide a clear comparison about it. The testing practitioners decompose the system based on the different use scenarios from which the formal representations are created and then the test scenarios are derived from these intermediate representations. Quality of these specifications is an important factor to be considered in order to generate an effective test data. Using semi-formal specification facilitates the test automation process but still, its cost expensive and not preferred much. Natural language-based representations are generally preferred in Requirements Engineering to facilitate easy communication among the stakeholders.

Furthermore, generating test scenarios in the initial phase of the development cycle provides more control on the coding and testing part which in turn may help in reduced cost and time. Thus, this proposed work will focus on generating test cases from requirements i.e. in the starting phase of the development itself.

In Agile process, User stories describe the functional requirements of the system written in Natural Language. The Natural Language Processing (NLP) tools helps in linguistic Analysis say, classifying the terms in to single part of speech which in turn can be developed to UML design. In Model-Based Testing, the test cases are generated at the early stages of development stages so that it is easy for identifying the inconsistencies and improve the specifications.

The second section discusses about literature survey and dimensions for research work in this field. Subsequent section describes the proposed algorithm for generating test data from user stories. Conclusions are given in the last section. This paper aims at better evaluation and comparison of test data generated from existing bio inspired approaches with the proposed method.

2. Related Work

AnnibalePanichella et.al proposed a Dynamic Many-Objective Sorting Algorithm (DynaMOSA) for addressing the test case generation problem in the context of coverage testing. [1] Covers three coverage criteria say branch coverage, statement coverage and strong mutation coverage and does not include non-coverage criteria like execution time and test size etc.

MeryemElalloui et.al proposed a transformation process (NLP) that converts the user stories into UML use case diagrams. [2] has features that includes pre-processing of user stories, parse tree construction for each user story and then extracts actors, use cases and relationships from it. Use case diagrams are built by applying NLP techniques. NLP analysis encounters a problem from complicated sentences. The technique helps the designer in clear interpretation of user stories that reduces the time to draw use case diagrams and improve the workflow.

Pardeep Kumar Arora et.al used the UML class diagram, use case and activity diagram for identifying changes at syntax and semantic level [3]. Agents are developed to collect the changes in distributed environment but did not address about the Formal specifications.

Ali Shahbazi et.al proposed an Adaptive Random String Test case generation (FCFS) and Evolutionary String Test Case Generation methods (Genetic Algorithm) [4] Focuses on string inputs rather than other test case types like trees or graphs.

Meiliana et.al [6] uses modified DFS algorithm that generates test cases from UML diagram at early stages of SDLC.A graph say System Testing Graph (SYTG) is designed and the test case data are stored into this graph. [7] translates automatically the Restricted-form of Natural Language requirements specified as Scenarios into executable Petri-Net models from which test scenarios are generated.

Kumar introduces a semi-automatic tool UMGAR [8] that generates UML models from natural language requirements using Natural Language Processing tools. UMGAR used syntactic reconstruction for translating the complex into simple requirements.

Shunkun Yang et.al proposed a search based method for generating test data [9]. Regeneration Genetic Algorithm (RGA) solves the population aging problem. RGA was efficient compared to the traditional GA and few other methods that are helpful in efficient testing with increasing number of test inputs. Deepak Kumar presented a survey on applying genetic algorithm for random generation of test cases and compared with various hybridized Genetic algorithm [10].

RashmiRekhaSahoo et.al proposed a distance based fitness function for generating test inputs for path coverage. [11] Uses search based approach PSO with ICF function Abdullah B. Nasser et.al proposed a Flower Pollination Algorithm [12] which reduced the size of test data thereby saving time and effort. [13] Introduced a modified Bacterial Foraging Technique (BFT) for solving the economic load dispatch problem. The randomness in the movement of bacterial foraging is overcome by heading the search towards promising locations based on the best previous values in the particle swarm optimization approach. The modified approach was able to handle high order cost polynomials with no additional cost or effort. Also did not require additional memory.

SaeedMotiian et.al proposed a novel algorithm [14] that generates the search faster than the standard Particle Swarm Optimization algorithm thereby minimizing the drawback of getting trapped at a local optimal solution during search. They used Hidden Markov Model (HMM) as the fitness function to find the minimum of the Ackley function.

Siva Suryanarayana et.al proposed a Flower Pollination Algorithm for the test case optimization in [15] with high fitness function. Test case optimization is mainly used in software testing for improving the fitness of test input generated. PriyankaDhareula et.al proposed genetically modified Flower Pollination algorithm (GM-FPA) [16].

3. Test Case Generation From User Stories

Research issues which motivated this work are: 1) lacking the ability to identify the major domain requirements 2) lack in generating an optimized test cases leading to fault recovery.

The proposed framework focuses on generating test cases from agile user stories. First, the requirements are captured in the form of user story and then processed with NLP from which actors are identified for generating the use case model. From the use case diagram the test cases are generated and optimized using the hybridized bio inspired algorithm.

3.1 Transformation of User Stories to Use Cases Using Nlp

In Agile development, User story capture the functional requirements in the below format “As a [role], I want [behavior], so that [business_value]".First, we convert the User storiesusing Natural Language processing techniques.

Features of the user story is parsed using the POS Tagger which retrieves the nouns and verbs which are matched with the test scenario descriptions we have already specified. If it doesn’t match, the lemma and synonyms of the search words can be used for comparison.

Test Scenario is the textual representation of the interaction between the system and the user. Tree Tagger parser and POS Tagger are used for preprocessing the user story. By applying Part Of Speech tags, the terms in

the test scenario are categorized into noun or adjective or verbs. Every item in the user story is converted into a single part of speech. After parsing, the nouns are taken as actors and verbs as use cases for generating the use case diagram using Java. ElementFactory Class is for creating actors and use cases. The ModelFactory class creates model, adds elements generating the UML file. The rules used are

Each first Noun, singular or plural Noun is an Actor.

All the Verbs or past participle are use case.

For each story an association relationship exists between the actor and the use case.

Algorithm 1:

1: Input: User Requirements as user story

2: Initialize Ex_ Actor with existing actors

3: Initialize currActor with current actor

4: Initialize PS with part-of-speech

5: Initialize isFst as True

6: Initialize isAvailable as False

7: if (((PS = = N)||((PS= = S_Noun)

|| (PS= = Comp_noun)))&&isFst)

8: for each (Actor Ac)

9: if (isAvailable == True) then

10: search actor

11: currActor= Ex_Actor

12: end for

13: if (isAvailable = False) then

14: Ac = createActor

15: if ((PS==Verb) ||(PS==pastparticiple))

16: V = add (tokn)

17: UseCase UC = ElementFactory.createUsecase(V)

18: Association asc = ElementFactory.createAssociation(currActor, UC)

Once the Actors and use cases are generated, the flows among the activities are identified and then the Dependency graph is generated. Vertices/Nodes are the use cases and the edges represent the dependencies between use cases.

3.2 Hybridized Bfa-Pso-Ga Based Test Case Generation and Optimization

A new hybridized evolutionary algorithm is proposed for generate optimized test cases that maximizes the fitness function f (λ) = log P(o/ λ). The fitness function helps in finding the bestoptimized test cases and helps in identifying the direction of search as well as magnitude of the iteration [25]. The Proposed method combines the Bacterial Foraging Algorithm (BFA) with Particle swarm optimization (PSO) and Genetic Algorithm (GA) to overcome the random movement of search, thereby search will be directed towards global optimum test cases without getting trapped at the local optimal solution.

Bacterial Foraging Algorithm (BFA) [24] evolved from the E.Coli bacteria's foraging mechanism which identifies the higher nutrient places thereby avoiding noxious places. BFA is applied for solving many optimization issues like optimal control, load balancing, harmonic estimation etc.

Chemo taxis is a foraging behavior which does the optimization process. From the current position, search will be directed to the position that has a minimum fit. The speed of the search can by influenced step size Cs. Continuing the search in the same direction is swimming and changing the direction of search is referred as tumbling. The direction after tumble is based on the position of every target and its velocity using Particle Swarm Optimization (PSO) technique [23][26][27][28]. The random movement of BFA is overcome by using the particle and swarm best values. At each increment i, velocity VL and position PS of each node are updated using (i) and (ii)

VLi= VL i-1 + C1 * rand()* (p_bst – PS i)+ C2 * rand( ) * (g_bst - PSi) (i)

"PS i = PS i-1+ VLi" (ii)

Initially velocity are chosen randomly generated within the range [-VLm, VLm] where VLm is the maximum value that can be assigned to any VLi. p_bst and g_bst are the best values of the particle and swarm. rand ( ) takes a value between 0 and 1. C1, C2 are the cognition and social components. Usually it takes a value 2.

Thus, the test cases are searched from one position to another based on the sum of current input and the step size and direction generated from PSO. Thus the search will be directed towards the global optimum test case every time.

If at target position, the fitness value is lower than the previous value then the target will swim with the step size Cs till it reaches the minimum fitness value but only for a certain number of steps, SN. After swimming, it tumbles. Then in reproduction stage, after CHN chemotactic steps, the fitness values of all the target test data are sorted in ascending order based on their accumulated cost function value.

Using the crossover mechanism of Genetic Algorithm (GA) [22], the test cases having highest TCfit is eliminated and the other test cases are taken as the parent for the next iteration. Two sets of parents are selected from the fittest group and cross over are done for creating a number of new test cases.

Append the parent test case (TCfit) and the newly created test cases to form the all possible set of test inputs.

After RN reproduction steps, the test cases with probability value 0 to 1 and lower than certain threshold value (Ped) are removed and those data having probability value higher than Ped keep their current position. After elimination and dispersal step, search will continue until maximum reproduction steps are achieved and then followed by other elimination and dispersal event till maximum EN events are achieved. A detailed pseudo code for hybrid bacterial foraging algorithm is given in Algorithm 2.

Algorithm2 Hybrid Algorithm
Input: Use Case Dependency Graph
Output: Optimized Test Cases

```
Initialize  CHN, RN, EN, SN, Cs, VL, PS
Elimination-Dispersal Step: while x <= EN
 x= x+1
Reproduction Step : while  y <= RN
      y= y+1
Chemotaxis Step : If  z <= CHN
   For each node  i
    While  (swm< SN )
swm= swm + 1
 Compute  VLi= VL i-1 + C1 * rand()* (p_bst– PS i)+ C2 * rand( ) *   (g_bst – PSi)
 Compute" PS i = PS i-1+ VLi "                    If (f ( PSi) > f(PSi-1) )
   Let  f ( PSi) = f(PSi-1)
       Compute  new  PSi
           Else
swm= SN
      End if
   End while
 Compute TCfit
Eliminate test cases with highest TCfit. Consider as the parent for next generation.
Cross over 2 parents from best fit to create number of  new test cases
Append the parent test case (TCfit) and the newly created test case.
          End for
         End if
     End while
Eliminate the test cases having  Ped
End while
```

4. Experimental Results and Analysis

The efficiency of the proposed algorithm is determined by applying on a dataset extracted from the case study online shopping and compared the manual and automatic generation of use cases and actors. Table 1 and Table 2 shows all the outputs achieved from the 100 user stories taken for the online shopping case study.

Table 1. Manual Evaluation of use cases and actors

Actors	Use Cases	Relationship
100	170	170

Table 2. Automatic Detection of use cases and actors

Actors	Use Cases	Relationship
99	165	165

Items identified by both plug-in and manual evaluation are taken as True Samples (TS) and the items identified by plugin and not by manual evaluation are False Positive (FP) and items identified manually but not by the plugin are considered as Negative Samples (NS).

Table 3. Accuracy of the proposed approach

Actors			Use Cases			Relationship		
TS	FP	NS	TS	FP	NS	TS	FP	NS
98	1	1	150	20	25	110	23	26
Precision P = 98%			P = 92%			P = 90%		
Recall R= 98%			R= 88%			R= 89%		

A good fitness function helps in finding the best solution closer to the optimal result and helps to calculate the direction of search of the optimized test input [25].

The fitness function taken is

$f(\lambda)=\log[P(o/\lambda)]$

The parameters used for the hybrid BFA algorithm are as follows:

Size =500

Chemotactic step Cs = 10

Swims SN = 20

Reproduction steps RN= 15

Elimination – dispersion steps EN =15

Probability =0.1

Acceleration constants = 2

Inertia weight =0.7

Velocity =100

The optimal solution helps in maximizing the fitness function f(λ) and the proposed hybrid BFA is implemented using MATLAB. Table 4 and Table 5 show the statistics of the test case generated and its transition coverage.

Table 4. Test Cases for Online Shopping

Algorithm	Iteration Taken	Average Test Case generated	Execution Time(sec)
BFA	6	89	39.102
PSO	5	74	35.64
BFA-PSO-GA	2	60	29.325

Table 5. Test Case Statistics for Online Shopping

SNo	Use Case	Optimized Test Cases	% Coverage
1	Login	13	96%
2	Select Product	15	100%
3	Add to Cart	3	97%
4	Select payment mode	9	100%
5	Make Payment	20	97%

Figure 1 shows the average number of test cases generated by BFA, PSO and the hybrid approach BFA-PSO-GA algorithm. The proposed algorithm generates the optimized number of test case for the scenario with minimum execution time.

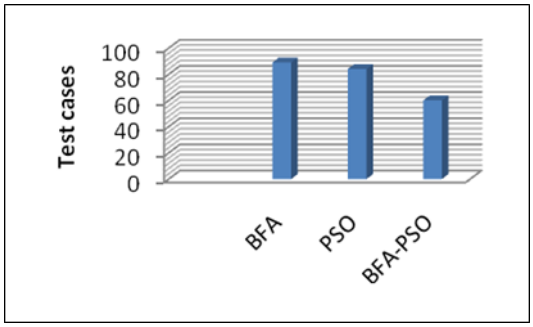

Fig.1. Comparison of Number of Test Cases Generated Using BFA, PSO and Proposed Algorithm

Figure 2 show that the proposed approach generates the optimized test data in a minimum number of iterations when compared with BFA and PSO. The random movement in the Chemotaxis step of BFA algorithm is reduced by taking the pbest and gbest values which leads to the optimized position.

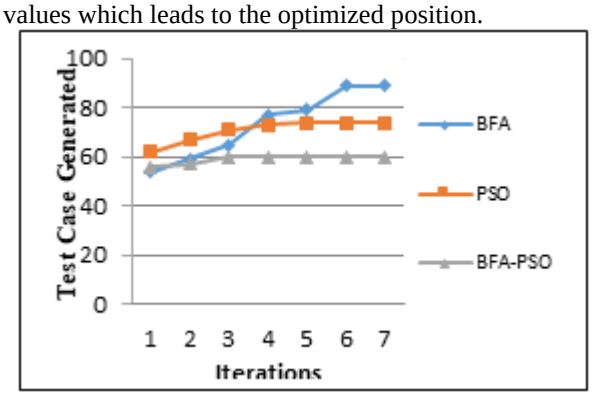

Fig. 2. Test Case Generated Vs. Iterations

The effort for test case generation is calculated using

EffortE=TimeTaken/ TestcasesGenerated

The number of test case generated for online shopping scenario is 60 and the time taken is 29.3 sec. For the feasibility study, the time taken for generation helps in determining which method is faster, manual or automatic. The effort spent on generating test cases is the average time spent for generating each test case. The proposed approach takes less time to generate test cases when compared with other algorithms. Also the effort required for each test case is lesser than the PSO and BFA algorithm. Less redundant test cases and test steps covering the boundary conditions helps in improving the quality of the test cases generated.

5. Conclusion

The proposed approach focus on capturing the user stories in an agile environment there by generating the UML model. Hybrid BFA-PSO-GA Algorithm generates the test cases from the Use cases. In the hybrid approach, the random movement of Chemotaxis step in BFA is reduced by finding the new position based on the previous best position pbest and gbest which in turn improves the convergence speed and also finds the global optimized test cases. The simulation results proved that the proposed approach generates the optimal test case

with highest test coverage at the early stage of the software development. Furthermore, our approach is more efficient than BFA, PSO and others as it generates the test cases in less iteration and with greater test coverage at a statistically robust level.

References

1. Annibale Panichella, Fitsum Meshesha Kifetew and Paolo Tonella " Automated Test Case Generation as a Many – Objective Optimization problem with dynamic selection of the Targets" in IEEE Transactions on Software Engineering, Vol.44, No.2, Feb 2018.
2. Meryem Elallaoui, Raja Touahni, Khalid Nafil, "Automatic Transformation of user stories into UML use case diagrams using NLP Techniques" , International Conference on Ambient System, Networks and Technologies, Procedia computer science 130:42-49, January 2018, DOI:10.1016/j.procs.2018.04.010.
3. Arora Pardeep, Bhatia, Rajesh, "Agent Based Regression Test Case Generation using Class Diagram, Use cases and Activity Diagram" International Conference on Smart Computing and Communications, Procedia Computer Science, 747-753,January2018,DOI:10.1016/j.procs.2017.12.096.
4. Ali Shahbazi , James Miller " Balck –Box String Test case Generation through Multi-Objective Optimization" in IEEE Transactions on Software Engineering, Vol.42, No.4, April 2016.
5. Jason Arbon, "AI for Software Testing", Pacific NW Software Qulaity Conference,2017.
6. Meiliana, Irwandhi Septian "Automated Test case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm", International Conference on Computer Science and Computational Intelligence, October 2017.
7. Edgar Srmiento, Julio, Leite S.P, "Test Scenario Generation from Natural Language Reguirements Descriptions based on Petri-Nets" in Electronic Notes In Theoretical Computer Science 329(2016) 123-148.
8. D.D Kumar, M.A Babar, " An Automated Tool For Generating UML Models From Natural Language Requirements", International Conference On Automated Software Engineering , pp 680-682. (2009).
9. Shunkun Yang, Tianlong Man, Jiaqi Xu, Fuping Zeng, Ke Li, "RGA: A Light Weight And Effective Regeneration Genetic Algorithm For Coverage Oriented Software Test Data Generation", Information And Software Technology 76(2016) 19-30.
10. Deepak Kumar, Manu Phogat, " Genetic Algorithm Approach For Test Case Generation Randomly: A Review", International Journal Of Computer Trends And Technology, Volume 49, No 4, July 2017.
11. Rashmi Rekha Sahoo, Mitrabinda Ray, "PSO Based Test Case Generation For Critical Path Using Improved Combined Fitness Function", Journal OfKing Saud University, Computer And Information Sciences, 32(2020) 479 – 490
12. Abdullah B Nasser, Abdul Rahman, Alsewari Nasser M Tairan, Kamal Z Zamli, " Pairwise Test Data Generation On Flower Pollination Algorithm", Malaysian Journal Of Computer Science,pp:242-257,Sep2017DOI:10.22452/mjcs.vol30no3.5.
13. A Y Saber, G K Venayagamoorthy, "Economic Load Dispatch Using Bacterial Foraging Optimization With Particle Swarm Optimization Based Evolution," Proceedings Of The IEEE Swarm Intelligence Symposium, Sep(2008).
14. Saeed Motiian And Hamid Soltanian Zadeh, "Improved Particle Swarm Optimization And Applications To Hidden Markov Model And Ackley Fnction" IEEE Conference On Communication, October 2011, DOI:10.1109/ICBNMT.2011.6155997.
15. Siva Suryanarayana C H,Satya Prakash Singh, "Flower Pollination Algorithm For Effective Test Case Optimization In Software Testing", International Journal Of Engineering And Technology, Volume 9, Issue-1, October 2019.
16. Priyanka Dhareula , Anta Ganpati , " Sofyware Test Case Prioritization Using Genetically Modified Flower Pollination Algorithm (Gm-FPA)", International Jour Nal Of Scientific And Technology Research,Volume 8, Issue 12, December 2019.
17. L V Xuewei, Huang Song Hui, Zhanwei Ji, Haijin, "Test Case Generation For Multiple Paths Based On PSO Algorithm With Metamorphic Relations", IET Software 2018, DOI: 10.1049/iet-sen.2017.0260.

18. M R Keyvanpour, Homayouni, Hsein Shirazee, " Automatic Software Test Cse Generation", Journal Of Software Engineering 5(3): 91-101, 2011, DOI: 10.3923/jse.2011.91.101.
19. Dalal Sandeep, Chhillar, Rajendar, " A Novel Technique For Generation Of Test Cases Based On Bee Colony Optimizatiomn And Modified Genetic Algorithm (BCOmGA)", International Journal Of Computer Applications, 68(19):12-16, April 2013, DOI:10.5120/11687-7359.
20. S S Bodiwala , D C Jinwala, "Optimizing Test Case Generation In Glass Box Testing Using Bio – Inspired Algorithms", International Conference On Software Engineering And Service Science(ICSESS), 2016,pp 40-44, doi: 10.1109/ICSESS.2016.7883012.
21. Noraida Tsmail, Rosziati Ibrahim, Noraini Ibrahim, "Automatic Generation of Test Cases from Use-case diagram", Interantional Conference on Electrical Engineering and Informatics, June 17-19,2007.
22. H. Haga , A.Suehiro, "Automatic test case generation based on Genetic Algorithm and Mutation Analysis",IEEE International Conference on Control System, Computing and Engineering, Penang, 2012, pp. 119-123, DOI: 10.1109/ICCSCE.2012.6487127.
23. S.Tiwari, K.K.Mishra, A.K. Mishra, "Test case Generation for modified code using a variant of Particle Swarm Optimization (PSO) Algorithm", 10th International conference on Information Technology:New Generations , Las vegas, 2013,pp 363-368, DOI: 10.1109/ITNG.2013.58.
24. A.Tamizharasi, J.Jasmine Selvathai,A.Kavi Priya, Maarlin R, Harinetha, "Energy Aware Heuristic Approach for Cluster Head Selection in Wireless Sensor Networks", Bulletin of Electrical Engineering and Informatics, Vol. 6,No. 1, March 2017, pp. 70-75, DOI:10.1159/eei.v6i 1.598.
25. A.Tamizharasi, A.Kavi Priya, "Heuristic Approach for Optimizing the localization of Wireless Sensor Networks", International Journal of Applied Engineering Research, Vol. 11, No. 4(2016), pp 2327-2331.
26. Murugan, S., Jeyalaksshmi, S., Mahalakshmi, B., Suseendran, G., Jabeen, T.N. and Manikandan, R., 2020. Comparison of ACO and PSO algorithm using energy consumption and load balancing in emerging MANET and VANET infrastructure. Journal of Critical Reviews, 7(9), p.2020.
27. Sampathkumar, A., Murugan, S., Sivaram, M., Sharma, V., Venkatachalam, K. and Kalimuthu, M., 2020. Advanced Energy Management System for Smart City Application Using the IoT. In Internet of Things in Smart Technologies for Sustainable Urban Development (pp. 185-194). Springer, Cham.
28. Sampathkumar, A., Murugan, S., Rastogi, R., Mishra, M.K., Malathy, S. and Manikandan, R., 2020. Energy Efficient ACPI and JEHDO Mechanism for IoT Device Energy Management in Healthcare. In Internet of Things in Smart Technologies for Sustainable Urban Development (pp. 131-140). Springer, Cham.