International Journal of Software Engineering and Knowledge Engineering

A Model Based Path Selection Testing on Mobile Apps using Tabu Monitored Hybrid-Local Search Optimization algorithm*

T. TAMILARASI†

SITE, VIT University, Vellore,
Tamilnadu,Vellore-632014, India‡
tamil.arasit2014@vit.ac.in§

M. PRASANNA

SITE, VIT University, Vellore,
Tamilnadu,Vellore-632014, India
prasanna.m@vit.ac.in

Abstract

Android Applications are the most widely used Apps for all types of real time applications like online shopping, chat apps and more Play store applications. The use of mobile devices and android apps are getting more. So, the need for testing the mobile apps is also more essential in order to improve app development in several criteria like risks, security and so on. Many of the researchers had much progress in automation testing of mobile apps but a key problem remains the same i.e. to determine the correctness in test case executions. This proposed work presents (i) mobile application design, (ii) Retrieving sequence diagram and Data flow graph from the Application model, (iii) Manual testing of application model using Tabu Monitored Hybrid Local search optimization algorithm, (iv) Automated testing of Mobile apps and (v) Test report generation. Using Tabu Monitored Hybrid Local search optimization algorithm, the mobile application is tested, followed by test report generation. Testing the mobile app using this algorithm is capable of detecting faults in loops, synchronization faults and the flow of path through nodes. Using this algorithm, the critical path is identified from the data flow graph. This proposed approach prioritizes the test path and hence the generated test cases get minimized.

Keywords: Automation testing, mobile apps, critical path, synchronization, flow graph.

1. Introduction:

Mobile applications are becoming one of the emerging technologies which become smarter and more powerful. Despite of the increase in the app development population, automation testing of mobile apps can improve efficiency in the field of android engineering. Due to the rapid changes and dramatic variations in app devices, software

test engineers must adapt quickly to the mobile environment. This framework describes (1) Mobile app development, (2) Manual testing of app using sequence diagram, (3) Automation testing of the developed application, and (4) Generating test cases for the mobile application.

Diane E Vaughan[1] et al. describes that the wide variety of local search algorithms make it difficult to evaluate the performance of such strategies. Tabu search strategies are Meta heuristics that guide local search algorithms to generate solutions where local search algorithm applied independently. This paper formulates the tabu search strategies that guides Generalised Hill Climbing algorithm. The algorithm is implemented with iterative dependent function in which each outer loop iteration is modelled as Markov Chains, inner loop iterations is performed with tabu search strategies. Each inner loop iterations are optimized to outer loop iteration with solution space and hence it is referred as Tabu Guided Generalised Hill Climbing Algorithm (TG^2HC).

2. Literature Review:

Ying dar lin et al[2] have did GUI testing with GUI tool with camera where it reduces the time required to record test cases and increases reusability of the test oracle without compromising test accuracy. Time consumption is the major problem of model based testing. In order to create a model, it is required having a detailed documentation for SUT (System under test) or else invalid test cases will get generated. SPAG here is nothing but Smart phone Automate GUI testing tool. This tests the android devices with record-replay technique, where the testers the test how the application supposed to work but SPAG tests and reports whatever necessary scenarios for testing. On the contrary, SPAG-C takes the advantage of the method SPAG which accurately tests the DUT (Device under test). SPAG depends on the image verification from android devices where SPAG-C performs image verification with the images captured in the camera. SPAG depends on android devices for snapshot, whereas SPAG-C inherits with the characteristics of SPAG and also improves the verification process by making accurate, reusable and better synchronized.

Chuanqi Tao et al [5] discuss test complexity evaluation methods for test environment. This is an approach for modelling mobile environments based on Mobile Test

Environment Semantic Tree (MTE_{ST}). To achieve effective test automation, test solutions must be compatible, deployable and executable on different mobile devices, platforms, network and applications. MTE_{ST} is used to assist engineers to perform test modelling and analysis for mobile test environments. Firstly, it uses a model based approach to modelling, presenting an analysis of diverse mobile test environments. Secondly, it provides a systematic method to evaluate test complexity of diverse mobile environment deployments. Thirdly, two realistic mobile apps are studied using proposed models. There is a lack of précised test models and coverage criteria to address distinct features and needs in mobile testing. This requires a special test model to address the test coverage criteria for complicated mobile apps.

Anbunathan R et al [6] describes in his paper that test automation framework helps engineers to write scripts and then execute those scripts in PC. The scripts are portable which support cross platforms, like windows and Linux with minimal changes in the executable statements. This paper discusses about automation test framework developed for functional testing of Android mobile devices. It includes use of cross platform scripting environment and using this, an automation testing tool has been developed called AutovalX. This tool can run both on windows and Linux platforms.

Features of AutovalX tool:

- Capture the image
- Compare the image
- Mask the image and
- Records the results in database.

Chanda Chouhan et al [7] proposed a testing model called TCBAD approach. This model generates test cases on the basis of the activity diagrams. Activity diagrams are used in representing the workflows of stepwise activity. From the activity diagram, activity dependency table gets generated where it shows the dependencies of each and every workflow by means of IN degree and OUT degree. Then, ADG (Activity dependency graph) gets generated from the dependency table. By implementing DFS (Depth First Search) test cases are generated from ADG. Finally, cyclomatic complexity is being calculated to find out the minimum number of test cases generated from the activity diagram.

Macario Polo et al [8] describe in his paper that testing as a destructive task where the goal is to find defects as early as possible. This article is the review of diverse technologies for test automation. Test automation reduces costs and increases the quality of the testing tasks.

XUnit frameworks are the most used technology to automate tests. In such frameworks, test cases are written in an executable language and can be executed automatically. They also provide specific operations to implement the test case oracles. Capture and replay tools comprise another commonly used technology; they register a tester's interactions with

the SUT, save them as a test script, and automatically re-execute them. Finally, specific test management tools are increasingly relevant for distributed software development and collaborating teams from different sites, or even different companies, or for efficiently connecting with test outsourcing providers.

3. Mobile Application Overview

3.1 Application 1-Student Result Automation System

This app is a developed application for Automation testing and comparative analysis with existing Google Play store apps. Student result automation system is an android application developed to automate the result publishing procedure of students. The result of a student is displayed only when the student enters his/her own roll number and name. The main aim of the system is to help the school or university exam officials in evaluating the student results and its disclosure. This application can be executed only in android app player or in any android application system devices. Android device, being more portable and flexible, can be used as educational tools. So, in order to enhance and automate the result disclosure process, Android Student Result Automation system is proposed, and it has a good future scope [29,31].

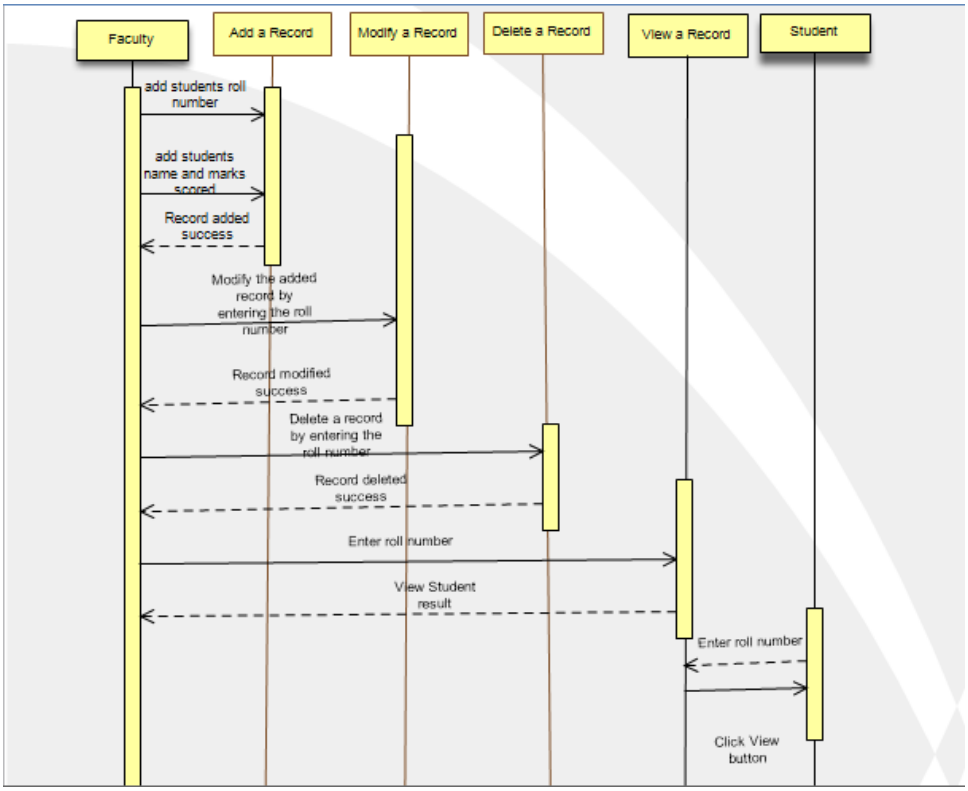
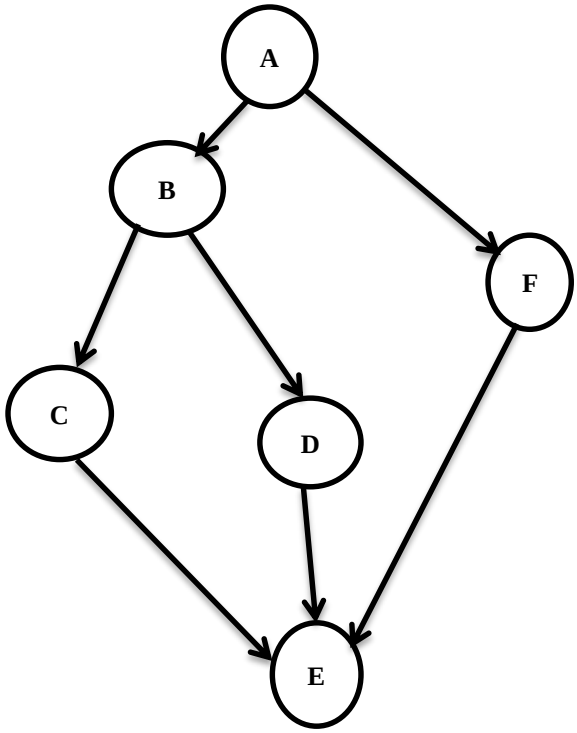


Fig. 1. Sequence diagram for student result automation system



A- Faculty

B- Add record(Roll no, name, marks)

C- Modify record

D- Delete record

E- View Record

F- Student

Fig. 2. Data Flow graph of Student Result Automation System

3.2 Application 2-Student Attendance Management System

This is an existing Google Play store application developed by Ali Ansari. Mark attendance app is a utility application used for marking attendances of students in Schools/Colleges/Universities [4].

Features and functionalities of this app [4,30]:

1. The user can easily add a subject.
2. User can add as many subjects as per the student's wants their subject.
3. Each subject will contain all attendance history of all students as well as for specific students showing all his past records.
4. User can change the date according to their will.
5. User can see a student's profile.
6. User can see a subject's details.
7. User can take attendance for a certain subject using attractive UI for specific date.
8. User can modify student's details.
9. User can modify subject's details.
10. User can upload an image as a profile image for students whether by capturing using camera or by choosing from gallery.
11. User can delete subject as well as subject details.
12. User can delete student as well as student attendance history.

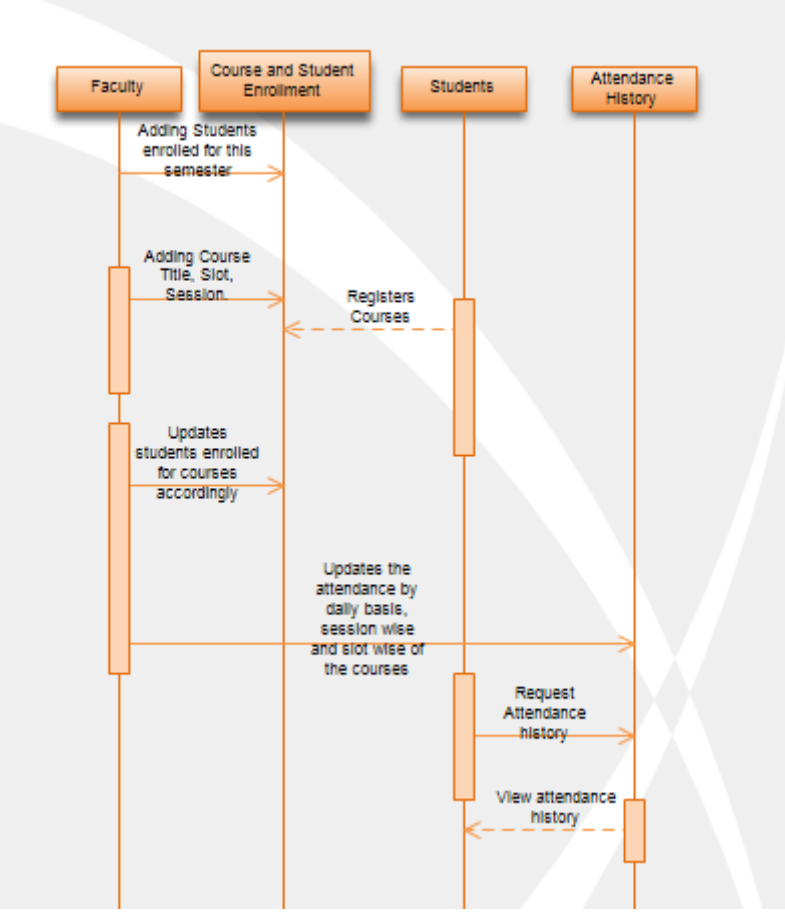
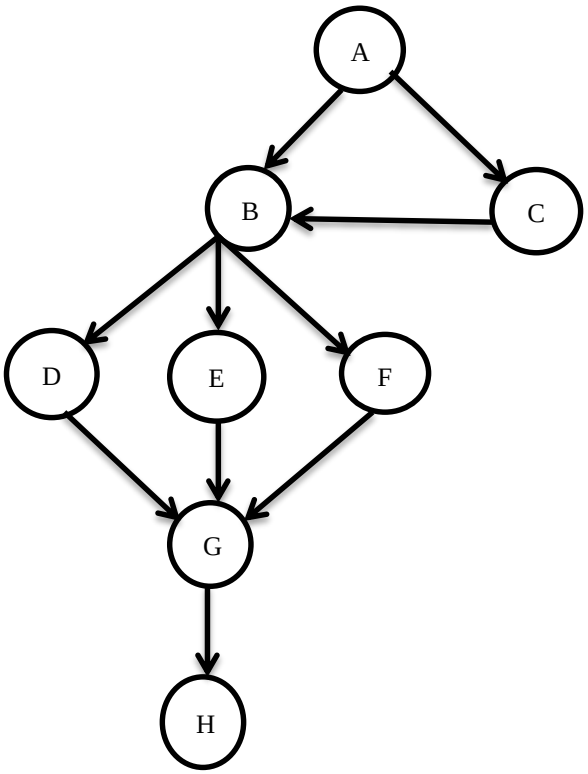


Fig. 3. Sequence diagram for Student Attendance Management System



A- Mark Attendance

B- Courses enrolled

C- Students enrolled

D- Students enrolled in Course 1

E- Students enrolled in Course 2

F- Students enrolled in Course 3

G- Mark Attendance

H- View Attendance History

Fig. 4. Data Flow Graph of Student Attendance Management System

4. Existing framework

4.1 Hill Climbing algorithm

The Hill Climbing algorithm is an Optimization algorithm and is a Local Optimization algorithm (contrasted to Global Optimization). It is a direct search technique, as it does not require derivatives of the search space. Hill climbing is a very simple idea which start from start state 'S' and move to a neighbour 't' with better score. The neighbourhood is the set of neighbour nodes connected with the Start or current node which is also called move set. The current state is the problem space which tends to have structures. A small change produces a neighbouring state in which that neighbourhood must be small enough for efficient search optimization. Designing the neighbourhood node is critical where the real ingenuity lies. Choosing the best node among many number of neighbour nodes is hard.

4.2 Tabu Search Algorithm

Tabu Search was devised by Fred Glover, is described as having four types of memory, Recency, Frequency, Quality and Influence. Recency and Frequency memory are lists of what has been tried and are used to guide the search to new areas of the search space. The meaning and use of quality memory and influence memory is less well defined. This describes Quality memory and Influence memory as being about more than cost (i.e. solution quality), although there does not appear to be a detailed description of what is meant by this.

Tabu Search uses a steepest ascent hill climber until it reaches a local optima pseudo code. It then implements the move with the smallest non-improving quality in the neighbourhood. It then makes some element of this move Tabu, so as to stop it being undone, this is stored in the Recency list. It then looks for and implements any improvements it can find. When there are no more non-Tabu improvements left in the neighbourhood it will repeat the process, implementing the smallest non-improving move, make it Tabu, and so on. A limit is imposed on how long move elements remain Tabu, this allows moves to be undone and redone.

To prevent the search looping Tabu Search uses a Frequency list to count how many times a move has been made Tabu and prevents the non-improving move from being made again if the count is too high. Frequency prevents improving moves from being undone and controls search width. Recency prevents recent non-improving moves from being undone and controls search depth. When Tabu Search uses steepest ascent it can be quite slow because it needs to evaluate the entire neighbourhood, candidate lists are one way of getting round this problem.

5. Implemented framework

5.1 Tabu Monitored Hybrid-Local Search Optimization algorithm

Applying Tabu Monitored Hybrid Local Search optimisation algorithm in Student result automation system:

1. Bp=A
2. Ns=B or F
3. If (Count==1)
   Bp=Ns
4. Else if(Count>1)
   If (fn (B)>fn (F))
   i. Bp=B
   ii. Else if(fn(F)>fn(B))
      1. Bp=F
   iii. Else
      1. Bp=B and F
5. Mov→C,D
6. If fn(C) =fn (D), then Bp=C and D.
7. Mov→E
8. Bp=(A-B-C-D-E)

Applying Tabu Monitored Hybrid Local Search optimisation algorithm in Student Attendance Management System:

1. Bp=A
2. Ns=B or C

    If (count==1)

        Bp=Ns

    Else if (count>1)

        If (fn (B)>fn(C))

            Bp=B

        Else if (fn(C)>fn (B))

            Bp=C

        Else if (fn (B) ==fn(C))

            Bp=B and C

    Break
3. Mov→D,E,F
4. If (fn(D)=fn(E)=fn(F) and Count=1), then Bp=D,E,F
5. Mov→H
6. Bp= prioritized path (A-B-E-G-H).

    A-B-E-G-H, A-C-F-G-H, A-C-B-D-G-H.

    Best path=A-B-E-G-H

Table 1 Algorithm Description Table

S.No	Notations	Description
1.	Bp	Bp=Best path
2.	Ns	Ns=Next Node
3.	Count	Count is the variable that tracks the total no. of solutions of the neighbour nodes
4.	Fn	fn=Fitness function, which calculates the no. of incoming nodes and the no. of outgoing nodes.
5.	A,B,C,D,E	Nodes in the data flow graph.

6. Test Report Generation

6.1 Dependency Tables

Table 2 Dependency Table for Student Result Automation System

Path Symbol	Test Scenario	Dependency	Input	Expected Output
A	Faculty			
B	Add record	A	Roll no, name and marks	Record added successfully.
C	Modify Record	B	Enter roll no and modify the record.	Record Modified.
D	Delete Record	B	Enter roll no and delete the record.	Record Deleted.
E	View Record	C,D	Enter roll no and view the details.	Record updated successfully.
F	Student	A	Enter roll no and view the details.	Record displays the marks of the specified roll no.

Table 3 Dependency Table for Student Attendance Management System

Path Symbol	**Test Scenario**	**Dependency**	**Input**	**Expected Output**
A	Admin			
B	Course enrolment	A,C	Add course title, slot, session, Add students enrolled.	Courses added.
C	Students enrolment	A	Add student name, course details enrolled, email, and phone number.	Students added and enrolled.
D	Students enrolled in course 1	B	View students enrolled and slot details of course1	Course1 details and enrolled students.
E	Students enrolled in course 2	B	View students enrolled and slot details of course 2	Course2 details and enrolled students.
F	Students enrolled in course 3	B	View students enrolled and slot details of course 3	Course3 details and enrolled students.
G	Mark attendance	D, E, F	Mark attendance of all the courses 1, 2 and 3	Attendance marked.
H	View Attendance	G	View Attendance History	Marked attendance is updated.

7. Test Results

Table 4 Test Case Generation for student result automation system

Test Path Number	**Test Path Node**	**Node Input**	**Node Output**	**Test Path Input**	**Test Path Output**
A	Start				
B	Start Add record Check the record added. End.	Register number, Name and Marks obtained by the particular student.	Record added success	Valid register number, valid name and marks obtained.	Record added.
C	Start Modify the record added. Check the record that is modified. End.	Enter register number and Modify the name or marks obtained by the particular student.	Record Modified success.	Valid register number to go for modification. Then valid name or marks to be modified.	Record Modified.
D	Start Delete the record added. Check the record whether it is deleted. End.	Enter register number and delete the entire record by clicking delete button.	Record Deleted Success.	Valid Register number to go for deletion. Then Click delete button.	Record Deleted.
E	Start View the record. Check the information while viewing the record. End.	Enter the register number and click view.	Record Viewed Success.	Valid register number and then click view.	Record Viewed.
F	Start Enter register number End	Enter register number	Record With name and marks obtained	Valid register number.	Record obtained.

Table 5 Test Case Generation for Student Attendance Management System

Test Path Number	**Test Path Node**	**Node Input**	**Node Output**	**Test Path Input**	**Test Path Output**
A	Start				
B	Start Add course details. Check the enrolled courses added. End.	Course Title, Class Number, Batch.	Course details added.	Enter valid course details.	Course details added.
C	Start Add student's details. Check the students enrolled which are added. End.	Add Name, Course title, Batch, Email and phone number.	Student details added.	Enter valid details of the students.	Student details added.
D	Start Add students enrolled in Course1. Check the added list. End.	Add Students enrolled in Course 1.	Course enrolled and students enrolled are added in Course1.	Enter the students enrolled for this Course1.	Course and students enrolled are added for course1.
E	Start Add Students enrolled in Course2. Check the added list. End.	Add Students enrolled in Course 2.	Course enrolled and students enrolled are added in Course2.	Enter the students enrolled for this Course2.	Course and students enrolled are added for course2.
F	Start Add Students enrolled in Course3. Check the added list. End.	Add Students enrolled in Course 3.	Course enrolled and students enrolled are added in Course3.	Enter the students enrolled for this Course3.	Course and students enrolled are added for course3.

Test Path Number	**Test Path Node**	**Node Input**	**Node Output**	**Test Path Input**	**Test Path Output**
G	Start Mark attendance for students enrolled in Course1, Course2 and Course3. End.	Mark Attendance for the enrolled students in course1, course2 and course3 by clicking Mark Attendance Button.	Attendance marked for students enrolled in Course1, Course2 and Course3.	Mark attendance for students enrolled.	Attendance is marked.
H	Start View attendance that is marked for all courses. End.	View attendance for the enrolled students by clicking the course type.	Attendance viewed for enrolled students.	View attendance for enrolled students.	Attendance is viewed.

7.1 Results Obtained:

By applying Tabu Monitored Hybrid Local Search Optimization Algorithm is implemented in the flow graphs thus generated. Executing this algorithm implemented in these case studies generates the critical path by obtaining the best fitness value and the best point values. The design, development and results obtained for the application framework has been uploaded in [29].

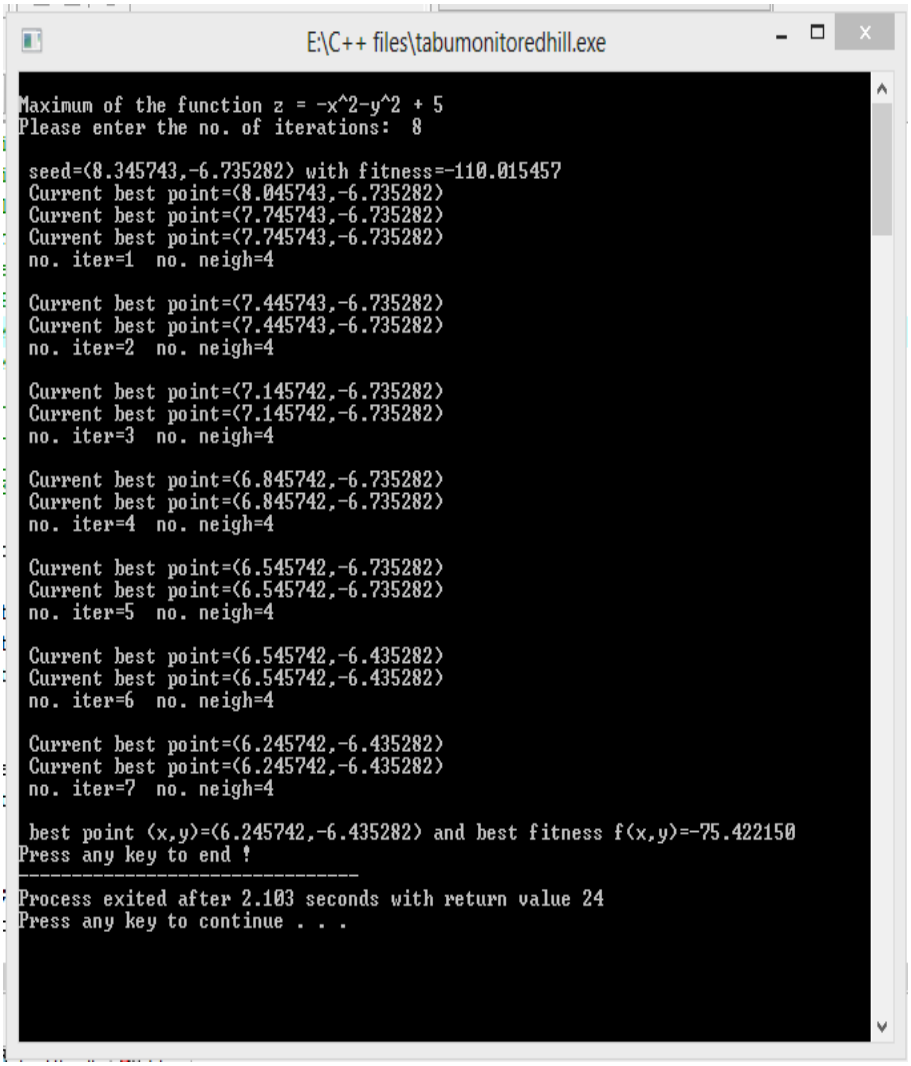


Fig. 6 Results obtained for the implemented framework

8. Conclusion and future work:

The Student Result Automation Project is an android app which can help in easy and fast result browsing from the database. The approach used is capable of detecting faults like faults in loops, synchronization faults, etc. Using this approach, the test path is prioritized. This Tabu Monitored Local search optimization algorithm combines the features of heuristics of hill climbing algorithm and tabu search algorithm which finds out the critical path for the data flow graph.

9. References:

1. Vaughan, D. E., & Jacobson, S. H. (2004). "Tabu guided generalized hill climbing algorithms." *Methodology and Computing in Applied Probability*, Volume 6, Issue 3, doi:10.1023/B:MCAP.0000026564.87435.66.
2. Ying- Dar lin., Jose F Rojas., Edward TH chu., Yuang Cheng Lai.(2014) "On the Accuracy, Efficiency, and Reusability of Automated Test Oracles for Android Devices", IEEE Transactions on Software Engineering, Volume 40, No. 10, October 2014.
3. Mush Honda, KMS Technology, '4 key challenges of mobile app testing', http://www.ministryoftesting.com/2014/01/4-key-challenges-mobile-testing/
4. Ali Ansari, 'Mark Attendance', https://play.google.com/store/apps/details?id=pro.manager.attendance&hl=en
5. Chuanqi Tao., Jerry Gao., "Modeling Mobile Application Test Platform and Environment: Testing Criteria and Complexity Analysis", ACM Digital Library, pp 28-33, 2014.
6. Anbunathan R., Anirban Basu., "Automation Framework for Testing Android Mobiles", International Journal of Computer Applications, Vol. 106, No.1,pp. 25-31, 0975-8887, November 2014.
7. Chanda Chouhan., Vivek Shrivastava., Parminder S Sodhi., "Test Case Generation based on Activity Diagram for Mobile Application", International Journal of Computer Applications, Vol. 57, No.23, November 2012.
8. Macario Polo., Pedro Reales., Mario Piattini., Christof Ebert., "Test Automation", IEEE Software, 0740 -7459, pp. 84-89, 2013.
9. H. C. Brandão de Oliveira and G. C. Vasconcelos, "A hybrid search method for the vehicle routing problem with time windows," Ann. Oper. Res., vol. 180, no. 1, pp. 125–144, 2010.
10. J. Poppenborg and S. Knust, "A flow-based tabu search algorithm for the RCPSP with transfer times," *OR Spectr.*, 2015.

11. J.-K. Hao, R. Dorne, and P. Galinier, "Tabu Search for Frequency Assignment in Mobile Radio Networks," *J. Heuristics*, vol. 4, pp. 47–62, 1998.
12. D. E. Vaughan and S. H. Jacobson, "Tabu guided generalized hill climbing algorithms," Methodology and Computing in Applied Probability, vol. 6, no. 3. pp. 343–354, 2004.
13. D. Xu, S. Member, W. Xu, S. Member, M. Kent, and L. Thomas, "An Automated Test Generation Technique for Software Quality Assurance," IEEE Transactions on Reliability, pp. 1–22, 2014.
14. R. Anbunathan, "Automation Framework for Testing Android Mobiles," International Journal of Computer Applications, vol. 106, no. 1, pp. 25–31, 2014.
15. Y. Lin, J. F. Rojas, E. T. Chu, and Y. Lai, "On the Accuracy , Efficiency , and Reusability of Automated Test Oracles for Android Devices," IEEE Transactions on Software Engineering, vol. 40, no. 10, pp. 957–970, 2014.
16. G. Gay, M. Staats, M. Whalen, S. Member, M. P. E. Heimdahl, and S. Member, "The Risks of Coverage-Directed Test Case Generation," IEEE Transactions on Software Engineering, vol. 41, no. 8, pp. 803–819, 2015.
17. E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The Oracle Problem in Software Testing : A Survey," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 507–525, 2015.
18. N. K. Gupta and M. K. Rohil, "Improving GA based Automated Test Data Generation Technique for Object Oriented Software," *IEEE Softw.*, pp. 249–253, 2012.
19. C. Chouhan, "Test Case Generation based on Activity Diagram for Mobile Application," *Int. J. Comput. Appl.*, vol. 57, no. 23, pp. 4–9, 2012.
20. M. Linares-v, "Enabling Testing of Android Apps," *IEEE Int. Conf. Softw. Eng.*, pp. 763–765, 2015.
21. G. D. C. Farto and A. T. Endo, "Evaluating the Model-Based Testing Approach in the Context of Mobile Applications," *Elsevier J. Electron. Notes Theor. Comput. Sci.*, vol. 314, pp. 3–21, 2015.
22. G. Hu, X. Yuan, Y. Tang, and J. Yang, "Efficiently , Effectively Detecting Mobile App Bugs with AppDoctor," *ACM Journal.*

23. S. Hao, B. Liu, S. Nath, W. G. J. Halfond, and R. Govindan, "PUMA : Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps Categories and Subject Descriptors," *ACM J.*, pp. 204–217, 2014.
24. H. Haga, "Automatic Test Case Generation based on Genetic Algorithm and Mutation Analysis," *IEEE Int. Conf. Control Syst. Comput. Eng.*, pp. 23–25, 2012.
25. C. Tao and J. Gao, "Modeling Mobile Application Test Platform and Environment : Testing Criteria and Complexity Analysis," *ACM J.*, pp. 28–33, 2014.
26. D. E. Vaughan and S. H. Jacobson, "Tabu guided generalized hill climbing algorithms," Springer Journal: Methodology and Computing in Applied Probability, vol. 6, no. 3. pp. 343–354, 2004.
27. J. Poppenborg and S. Knust, "A flow-based tabu search algorithm for the RCPSP with transfer times," Springer J. OR Spectr., 2015.
28. B. Cao, F. Glover, and C. Rego, "A tabu search algorithm for cohesive clustering problems," Springer J. J. Heuristics, vol. 21, no. 4, pp. 457–477, 2015.
29. Blogger.com, http://mymobileapptesting.blogspot.in/2016/07/student-result-automation-system.html
30. Blogger.com, http://mymobileapptesting.blogspot.in/2016/07/student-attendance-management-system.html
31. Blogger.com, http://mymobileapptesting.blogspot.in/2016/07/student-result-automation-code.html