Self-Tuning Flowcharts: A Priority-Based Approach to Optimize Diagnostic Flowcharts

Girish Bekaroo
Middlesex University (Mauritius Branch Campus)
Bonne Terre, Vacoas, Mauritius.
g.bekaroo@mdx.ac.mu

Paul Warren
Independent Researcher,
neuralwarp@gmail.com

Abstract— **Flowcharts have been used in problem diagnosis for a long time because of their effectiveness during process representation. However, with time, diagnostic flowcharts can become unmanageably complex and incomprehensible, thus leading to longer decision paths. A lengthy decision path also implies a time consuming diagnosis process while at the same time being boring to end users utilizing systems containing diagnostic flowcharts. This study investigates the extent to which diagnostic flowcharts can be made dynamic so as to optimize the decision making process without reducing the number of nodes. In this endeavor, the Dynamic Flowchart Parser Algorithm has been proposed using a priority-based approach to optimize diagnostic flowcharts within a diagnostic tool named Self Tuning Flowcharts.**

Keywords— Flowchart Optimization, Diagnostic Flowchart, Dynamic Flowchart Parser Algorithm, Problem Diagnosis.

I. INTRODUCTION

Problem diagnosis, which refers to identifying the nature of a problem by examining its symptoms, is often considered as a challenging task because it traditionally requires the knowledge of an expert in the relevant field (e.g. medicine, engineering, etc.) [1]. In the process of traditional problem diagnosis, an initial description of the problem occurred is needed, which an expert uses so as to analyze and identify the cause of the problem and finally apply a remedy to solve the problem [2]. One common approach to help problem diagnosis in a faulty system is via the use of diagnostic flowcharts [3]. Due to its simplicity while also being a good way to document knowledge developed over time, this approach has been commonly used for diagnostic decision making of medical related problems [4] and computer and electronic problems [5], among others. Even though diagnostic flowcharts have been widely adopted, a major problem observed is related to their maintenance. For every discovery of a new fault in the same problem domain, nodes need to be added within the same diagnostic flowchart. As such, the flowchart can become unmanageably complex and incomprehensible with time thus leading to lengthy decision paths [3]. In other words, users are confronted with many questions before reaching the solution.

Taking cognizance of this problem, Beygelzimer et al proposed an algorithm called GREEDY, which attempts to optimize diagnostic flowcharts based on a generated dependency matrix using the Bayesian network representation of such flowcharts [3]. Although the proposed approach showed to optimize diagnostic flowcharts, reduction in the number of decision nodes was observed and this might result in loss of important information. Hence, there is a need for another optimization approach during which important information is not lost from node elimination. Such approach can to optimize decision making process within diagnostic systems and robots [6]. As such, this paper investigates the extent to which diagnostic flowcharts can be made dynamic so as to optimize the decision making process without eliminating nodes from the original flowchart.

II. DIAGNOSTICS FLOWCHARTS

A flowchart can be defined as "*a diagram that shows the connections between different stages of a process or parts of a system*" [7]. It consists of a set of symbols and connecting lines that shows stepwise progression through a procedure, a process or a system. Unlike other types of flowcharts including system and program flowcharts, diagnostic flowcharts tend to begin directly with a decision node or sometimes, the start node may even be present but it has to be followed by a decision node. Furthermore, diagnostic flowcharts normally have more than one ending point as compared to system or program flowcharts. Moreover, in most diagnostic flowcharts, it can be observed that the upper level questions in the flowchart tend to be more general so as to identify symptoms of a problem from a broader view. While going deeper into the structure, the questions become narrower in order to eliminate less general symptoms whereby paving the way to the most appropriate solution. An example of a diagnostic flowchart used in computer repair is given in Fig. 1.

In the diagnostic flowchart given in Fig. 1., 3 types of nodes could be found, namely, a start node, 5 decisions nodes which are represented by the diamond shape and 6 solution nodes which are in turn represented by the rectangular shape. Furthermore, a decision path is referred to as any path taken from the start to a particular solution node.

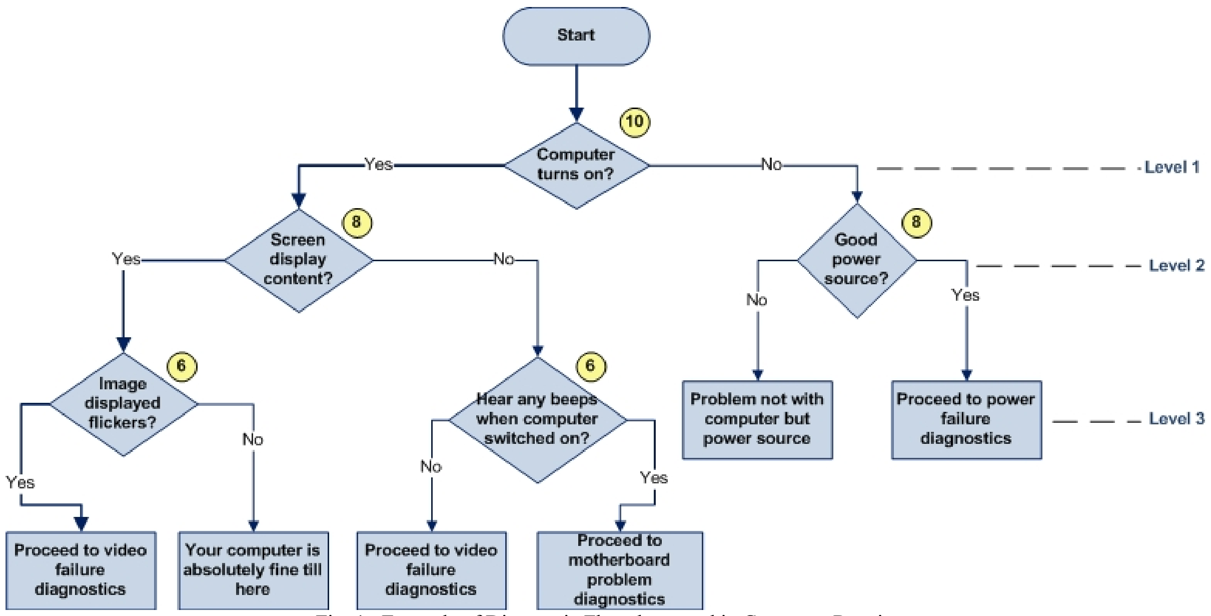

Start
Computer turns on?
10
Yes
No
Level 1
Screen display content?
8
Good power source?
8
Level 2
Yes
No
No
Yes
Image displayed flickers?
6
Hear any beeps when computer switched on?
6
Problem not with computer but power source
Proceed to power failure diagnostics
Level 3
Yes
No
No
Yes
Proceed to video failure diagnostics
Your computer is absolutely fine till here
Proceed to video failure diagnostics
Proceed to motherboard problem diagnostics

Fig. 1. Example of Diagnostic Flowchart used in Computer Repair.

III. Optimization of Diagnostic Flowcharts

An optimized flowchart is one that has a reduced average cost of diagnosis. Via the use of optimized diagnostic flowcharts, end users are expected to experience minimal average number of questions, thus also reducing the decision path length. To address this issue, a new algorithm named the Dynamic Flowchart Parser Algorithm (DFPA) was designed. The proposed algorithm is based on dynamic Huffman codes approach towards minimum redundancy tree [7]. DFPA aims to shorten diagnostic decision path by making the flowchart self-tuning itself, to fulfill the following requirements:

R1. Change the flowchart structure dynamically without elimination of nodes from the original structure,
R2. Make diagnostic flowcharts intelligent by learning from previous diagnostic paths,
R3. Reduce the mean cost of traversing the flowchart until reaching the solution while ensuring the accuracy of solution reached remains unaffected.

R1 addresses 2 essential points. Firstly, the flowchart structure should change dynamically, meaning that the decision path to reach a particular solution might vary during different parses of the flowchart. Secondly, no nodes should be eliminated from the original structure. Here, a node is said to be eliminated if and only if the node can never be part of the decision path whenever the flowchart is being parsed. R2 attempts to embed a learning process within such structures so as to be able to keep track of previous diagnostic paths. Finally, R3 is a major objective of DFPA towards diagnostic flowchart optimization while ensuring that accuracy of any particular solution is not compromised.

In order to meet its requirements, DFPA utilizes a priority based approach in addition to a learning algorithm based on path history. Every mode in the diagnostic flowchart is associated with a priority value where 0 means lowest priority and the highest number (n) meaning the highest priority. Every time the flowchart is parsed by a user, the priority of the decision nodes are affected based on the decision path adopted by that user. Based on these two approaches, every time the flowchart is parsed during diagnostic decision making, DFPA re-structures the flowchart based on the priority values of the nodes such that the flowchart appears to be self-tuning, while at the same time bringing the user closer to the decision. To achieve its purpose, the DFPA operates using 3 functions, described as follows:

A. *Initial Assignment of Priorities*

This function is conducted only once for every diagnostic flowchart and takes place the first time that the flowchart is used for diagnostic decision making. In this endeavor, DFPA assigns an initial priority value to every decision node present in the structure. For this, DFPA uses a formula based on the different levels along with the total number of nodes present in the flowchart. The idea here is to initially preserve the original flowchart structure before use for diagnosis. DFPA searches for the first decision node just after the start node of the flowchart and assigns it a maximum priority. DFPA then iteratively continues to parse the original flowchart while at the same time assigning the next level decision nodes with a slightly lower priority value than that of the previous level(s). As such, all the decision nodes on the same level are assigned the same priority. The formula used by DFPA is as follows:

*Priority value= (total number of decision nodes in flowchart * 2) – ((depth level of node-1) * 2)* (1)

As an example, the initial priorities assigned by DFPA to the flowchart from Fig. 1. are shown in the numbered circles. The levels of the decision nodes are also included in the diagram, shown by dotted lines, which are part of the calculation process. As shown in Fig. 1, the initial priority of the nodes remain such a way that the highest level in the flowchart takes the maximum priority in the flowchart whilst the lowest level taking the lowest priority thus preserving the initial structure of the original flowchart.

B. *Learning Process*

Once decision nodes have been assigned priority values, the structure can be used for diagnostic decision making by users. For this function, system behavior information via event traces [2] was adopted and adapted since such approach showed to be successful in problem diagnosis. In the learning process by DFPA, every time a user answers a particular question in the flowchart, the route taken by the same user is memorized. In this process, the priority values of nodes within the decision path are increased by 2 and that the remaining nodes outside the decision path is decreased by 2 by DFPA. It should be noted that the decision node just after the start node (i.e. Level 1 decision node in Fig. 1) is never affected in terms of priority value throughout the learning process by DFPA. For example, the priority value of the node "Computer turns on?" in Fig. 1 will always remain 10 throughout the learning process. Also, another rule is that the priority value of a lower level node can never exceed the priority of the node one level above it. By using this logic, lower level flowcharts cannot become higher than its upper level nodes so as to prevent the flowchart from becoming too dynamic which could affect the accuracy of the solution reached.

To illustrate how DFPA handles its flowchart learning function, consider the case where the user takes the highlighted route in Fig. 1 once. The priority values for the nodes "Screen display content?" and "Image displayed flickers?" will both increase to become 10 and 8 respectively. In contrast, the decision nodes outside the decision path, that is, "Good power source?" and "Hear any beeps when computer is switched on?" will have a decrease in their priority values to become 6 and 4 respectively. Also, as mentioned earlier, the priority value of the decision node after the start node remains 10.

C. *Dynamic Flowchart Parsing*

The third and most important function of DFPA is to dynamically parse the priority-based flowcharts in order to fulfill R3 and is triggered before jumping to the next node during problem diagnosis. Unlike traditional flowchart parsing, DFPA parses the flowchart based on currently assigned node priority values and during this process, the algorithm has to first decide on one of the following:

1. Has problem diagnosis just started and whether the first decision node to be answered is being searched?
2. Has a decision node already been parsed and whether the next node in the structure (either decision or an action) is being sought?

For the first case, DFPA searches for the decision node with the highest priority for display to the user. In case there are two or more such nodes with the same priority, the node from the deepest level is chosen. However for the second case, every time DFPA searches for the next node, it ensures that the following criteria are met:

- *A solution node has not been encountered*
  If a solution node is encountered as next one, it means that a solution has been reached and that the diagnosis process is complete.

- *Node found on logical path*
  Two nodes are said to be on a valid logical path if and only if a direct link exists between them. For example, nodes "Image displayed flickers?" and "Computer turns on?" are on the same logical path in Fig. 1. while nodes "Image displayed flickers?" and "Good power source?" are not on the same logical path.

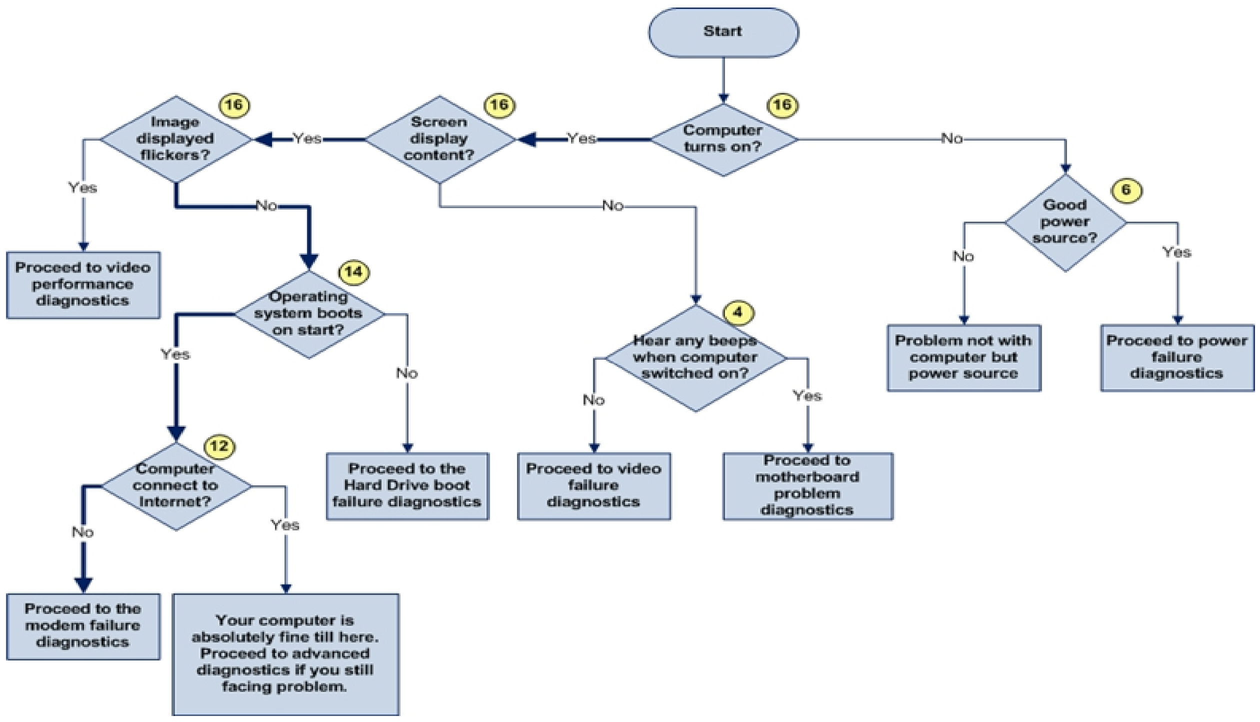

Start
Computer turns on?
16
Yes
No
Screen display content?
16
Image displayed flickers?
16
Good power source?
6
Proceed to video performance diagnostics
Operating system boots on start?
14
Hear any beeps when computer switched on?
4
Problem not with computer but power source
Proceed to power failure diagnostics
Computer connect to Internet?
12
Proceed to the Hard Drive boot failure diagnostics
Proceed to video failure diagnostics
Proceed to motherboard problem diagnostics
Proceed to the modem failure diagnostics
Your computer is absolutely fine till here. Proceed to advanced diagnostics if you still facing problem.

Fig. 2. Illustration of Dynamic Flowchart Parsing function.

As long as these criteria are met, DFPA can iteratively search for the next node based on the historical trail, until a solution is reached. To illustrate how this function works, consider the flowchart in Fig. 2. to diagnose Internet connectivity problem. This flowchart has already been parsed several times and the updated priority values of the nodes are encircled in the diagram. The most visited path is also shown by the thick line in the figure. Considering the current structure of the flowchart in Fig. 2., if the same path shown by the thick line was adopted by the next user, the order of the nodes parsed by DFPA along with simulated answers of a user would be as follows:

Node 1: *Image displayed flickers?*
User Answer: *No*

Node 2: *Computer turns on?*
User answer: *Yes*

Node 3: *Operating system boots on start?*
User answer: *Yes*

Node 4: *Computer connects to Internet?*
User answer: *No*

Node 5: *Proceed to the modem failure diagnostics* ***← (solution)***

Since the first node generated during the diagnosis is the one with highest priority but with deepest level, “Image displayed flickers?” is displayed as first node. The answer input by the user for this node is stored temporarily in memory. After the first node, for every 3 nodes parsed, DFPA jumps back to the originally skipped questions with the highest level first so as to cater for rare paths visited by users, thus preventing the rare paths from becoming too long. The number 3 here is a threshold variable for iteratively jumping to skipped nodes and throughout this study, the value 3 was maintained as threshold test value. As the second node, the skipped question, “Computer turns on?” is displayed and its answer input by the user is stored in memory. The two answers stored are then evaluated to get the next node by Dynamic Flowchart Parsing function of DFPA. Here, if the answer for the “Image displayed flickers?” node is “No” and the answer for the “Computer turns on?” is “Yes”, the next node returned by DFPA is “Operating System boots on start?” since both the first and second node answered are on the same logical path in the flowchart. Alternatively, if the answer for the “Image displayed flickers?” node is “Yes” and the answer for the “Computer turns on?” is “No”, the next node returned by DFPA is “Good power source?” since the logical path criteria is not met in this case. Here, an increase in path length by 1 node would be observed. This is also called a rare path and to cater for this, DFPA jumps back to skipped nodes after every 3 nodes, preset by the threshold value.

Continuing with the most visited path in Fig. 2., DFPA keeps on generating the next nodes while considering the defined criteria validated by the answer(s) from the user. Hence, in this case, a solution is reached after the fourth node generated by DFPA. This shows a reduction in the path length (4 nodes parsed instead of 5 in the original version) since one skipped question has not been parsed by DFPA. In other words, the node “Screen displays content?” was not considered and this node did not affect the accuracy of the solution as well. This is because, this node is found on an upper level, meaning that at the start of the diagnosis process, the elimination of symptoms starts from a wider area (more general questions) and converges to a solution. As such, skipping general questions do not affect the accuracy of the solution.

IV. EVALUATION

The proposed DFPA algorithm was evaluated so as to validate whether the algorithm has met its requirements (R1 to R3). For the identification of metrics needed to validate each requirement, the Goal Question Metric (GQM) approach was used [8] where different questions regarding the objectives of the algorithm were asked so as to obtain the correct metrics.

For R1, two important points need to be validated. Firstly, the dynamic change of flowchart structure should be verified and secondly, there should be no elimination of nodes from the original structure. For the first point, every DFPA parsed flowchart need to be checked so as to verify whether the decision paths are identical to the original flowchart. Within any parse, if the decision path of the DFPA parsed flowchart is different as compared to the original flowchart, the DFPA parsed flowchart can be considered to be dynamic. For validating the second point, a check for every DFPA parsed flowchart is needed so as to verify whether nodes from the original structure have been eliminated, based on the definition of elimination of node given earlier. As such, validating both points necessitate a comparison between the DFPA parsed flowchart and the original flowchart. Hereafter in this paper, the DFPA parsed flowchart will be called dynamic flowchart and the original flowchart will be termed static flowchart.

As discussed earlier, R2 attempts to embed a learning process within diagnostic flowcharts so as to be able to keep track of previous diagnostic paths. As such, assessing whether R2 has been successfully met needs a comparison against priority values between two successive flowchart parses.

To validate R3, the path length needs to be measured and compared against the original path length leading to the same solution. Since a solution path is equal to the number of decision nodes in that path, the metric for the evaluation of the dynamic flowchart approach is a count of the number of decision nodes needed to reach a solution in both static and dynamic flowcharts. This metric will be in the form of a numeric value ranging between 0 and the total number of decision nodes for the deepest path in the diagnostic flowchart. Another metric that could be used for this evaluation is the time taken to reach a solution. However, different users might take different amount of time within the same decision path due to various factors including computer literacy and experience with diagnostic flowcharts, among others. As such, this metric would not be as accurate as the node count.

Although assessment of R1 to R3 could be manually conducted, a diagnostic tool containing an implemented version of the DFPA was developed using Java and the tool was named Self Tuning Flowcharts (STF). A software approach was preferred to the manual approach in order to benefit from the various advantages for its use in evaluation namely, accuracy of results, reliability and accountability, among others [9]. STF contained downloaded diagnostic flowcharts related to computer repair [10] and although the flowcharts were manually stored with-in the database used by the prototype, techniques to visually recognize flowcharts nodes are presently available [11]. Such visual recognition techniques could be used in larger scale deployment of DFPA.

V. EXPERIMENTATION PROCEDURES

In the preparatory phase of the experiment, a randomly chosen diagnostic flowchart as in Fig. 3 was used, with the initial priority values assigned by DFPA. As the application area of STF was computer repair, the experiment targeted computer literate users who would be able to diagnose a simulated computer problem with the use of developed prototype. University students who utilize computers daily were targeted. Moreover, it was preferred that the participants were Computer Science students due to their knowledge in computer troubleshooting. Such profile would be able to effectively make use of the developed diagnostic tool to repair the simulated faulty computer by using their own knowledge and experience in solving the problem. Furthermore, the experiment needed several iterations so as to allow DFPA to gain enough path histories while contributing to its learning process. For the experiment, 16 users meeting the participant profile requirements agreed to participate and this number was enough to test the flowchart in Fig. 3. which contains 14 decision nodes.

To begin the experiment, every participant was introduced to the research and approval was sought via an Ethical Approval Form. Then, by making use of STF, the participant had to diagnose a simulated faulty computer without any other support. During this process, the participant had to check the faulty computer so as to find the answers to the questions generated by STF. While the participant was diagnosing the faulty computer, details on the behavior of the participant and any remarks made were observed and noted. When a solution to the problem was found by the diagnostic tool, the solution path from STF was noted, along with the path on the original flowchart. Then, with the support of the research team, the participant was made to fix the problem manually. The same process was repeated with the 16 participants with randomly simulated faults.

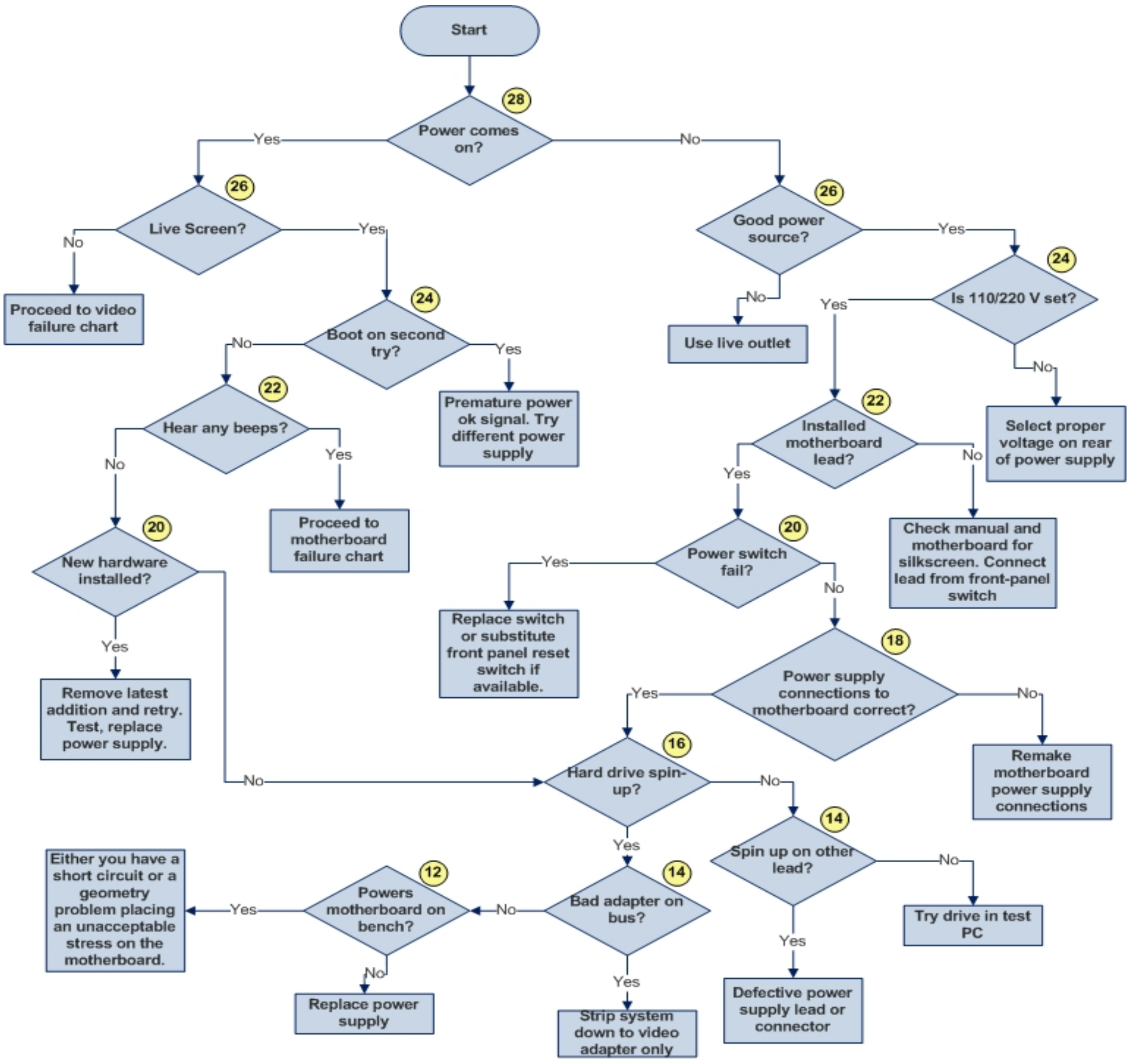
VI. RESULTS AND DISCUSSION

The main metric used for assessing R3 was path length, which is shown in Table I for both the dynamic flowchart parsed by DFPA and the original static flowchart for the different runs recorded during the experiment.

TABLE I. RESULTS

Run	Dynamic Flowchart Path Length	Static Flowchart Path Length
1	4	4
2	3	3
3	4	4
4	4	3
5	3	3
6	4	4
7	2	3
8	3	4
9	2	3
10	4	3
11	4	4
12	2	4
13	4	3
14	3	3
15	3	4
16	2	3

When the experiment started, it was observed that the number of nodes for both the static flowchart and DFPA were the same until the fourth run. This is because DFPA needed a few runs for learning path histories from previous participants. The first change in the structure of the flowchart was observed during the third instance of the test. A node from Level 2 was displayed first because it reached the same priority as the only Level 1 node in the flowchart. Hence, the Level 2 node had preference over the Level 1 node since it was at a deeper level. However, even though there was a change in the structure, the total number of nodes parsed was still the same in both approaches. This was because the skipped question was treated just after the Level 2 node. As mentioned earlier, the first difference in path length between the two approaches was noted on the fourth instance of the test and this was because the fourth participant visited a rare path. Consequently, one more question had to be answered by the user, thus making the dynamic flowchart longer than the original flowchart. The change in priority values of the nodes within the different runs also implies that the flowchart was affected by the answers given by the end user, thus contributing to the flowchart learning process. This also implies R2 was met.



Start
28
Power comes on?
Yes
No
26
Live Screen?
No
Yes
Proceed to video failure chart
24
Boot on second try?
No
Yes
Premature power ok signal. Try different power supply
22
Hear any beeps?
No
Yes
Proceed to motherboard failure chart
20
New hardware installed?
Yes
No
Remove latest addition and retry. Test, replace power supply.
26
Good power source?
Yes
No
Use live outlet
24
Is 110/220 V set?
Yes
No
Select proper voltage on rear of power supply
22
Installed motherboard lead?
Yes
No
Check manual and motherboard for silkscreen. Connect lead from front-panel switch
20
Power switch fail?
Yes
No
Replace switch or substitute front panel reset switch if available.
18
Power supply connections to motherboard correct?
Yes
No
Remake motherboard power supply connections
16
Hard drive spin-up?
Yes
No
14
Spin up on other lead?
No
Yes
Try drive in test PC
Defective power supply lead or connector
14
Bad adapter on bus?
Yes
No
Strip system down to video adapter only
12
Powers motherboard on bench?
Yes
No
Either you have a short circuit or a geometry problem placing an unacceptable stress on the motherboard.
Replace power supply

Fig. 3. Diagnostic flowchart used in experiment

For further comparison of both parsing methods, the node difference from each run was calculated by subtracting the total number of nodes parsed in the dynamic flowchart from the total number of nodes parsed in the static flowchart for reaching the same solution. A positive node difference in the graph means that the DFPA parsed lesser nodes than that needed in the static flowchart approach whilst a negative value means the reverse. The node difference for each run is depicted in the line graph in Fig. 4.

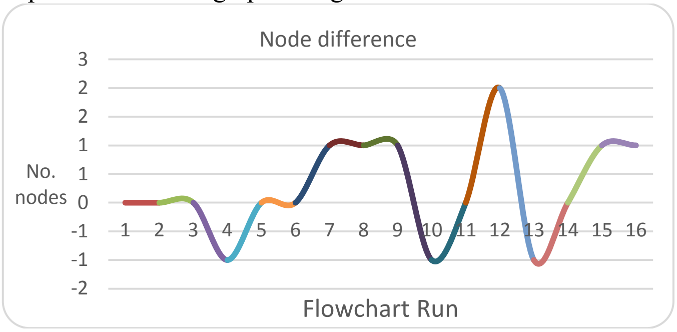


Fig.4. Node Difference.

In Fig. 4, it can be seen that the decision path after Participant 6 was starting to favor the dynamic flowchart approach. This is because DFPA had to learn from enough path histories. The most common difference in nodes was 1 (approximately 25% reduction) because of frequent visits of rare paths by users which also meant that the flowchart needed more parses to learn and get back to the path of frequently visited nodes. At most, 2 nodes were skipped by DFPA, but this did not affect the accuracy of the answer since the skipped nodes were of upper levels whilst the decision was on a deeper level. Overall, the total number of experiment instances where the dynamic flowchart gave a positive node difference was 6 (37.5%) as compared to 3 instances (18.8%) where the same type of flowchart gave a negative node difference. There was also the case where no difference in nodes was registered and this occurred 7 times (43.7%).

Results showed that dynamic flowcharts were better than its static counterparts since for 81.2% of the runs, the path length for dynamic flowcharts shorter than or equal to the path for the same solution in static flowcharts. This also confirms meeting the third requirement of DFPA for most cases. However, there were also some cases where rare paths were adopted by users which increased the overall path length. But, these cases happened to 18.8% of the cases and the number of were only increased by 1. Also, to cater for this problem, it was observed that DFPA re-structured flowchart towards the original during the diagnostic decision making session. This acts as a confirmation thus giving the chance for the rare node to gain some priority. The likelihood of the occurrence of rare paths also implies that any node could be parsed thus negating the likelihood for the elimination of nodes. The self-tuning nature of the dynamic flowchart in addition to no elimination of nodes from the original structure implies the second requirement of DFPA was also met.

Overall, all requirements of DFPA were met and the experiment also helped to identify various strengths of using the priority-based approach in the optimization of diagnostic flowcharts. Firstly, higher occurrence of shortened decision paths led to lesser time taken during problem diagnosis. Shortened decision path could facilitate the adoption of diagnostic flowcharts among practitioners. Also, embedding intelligence within dynamic flowcharts gave indications on the frequency of path adoption. This information could also be practically used for statistical analysis. Another notable advantage of DFPA was flowchart optimization without elimination of nodes from the original structure, thus avoiding loss of important information during the optimization process. On the other hand, the major weaknesses of DFPA were that rare decision paths can make the diagnosis longer and historical trails are needed for the learning function of DFPA.

VII. Concluding Remarks

This paper investigated the extent to which diagnostic flowcharts can be made dynamic so as to optimize the decision path without node elimination, by proposing an algorithm called Dynamic Flowchart Parser Algorithm. DFPA utilizes a priority-based approach for flowchart optimization and operates in three phases, namely, initial assignment of priorities, learning and dynamic flowchart parsing. An experiment was conducted so as to evaluate whether the proposed diagnostic flowchart optimization approach met its requirements. Out of the 16 participants of the experiment, 81% of the instances showed that the decision path was equal or shorter than the same path on the original static flowchart. As such, dynamic flowcharts parsed by DFPA were found to be better more optimized than their static versions for the most frequently visited path by users. This implies that in most cases, the dynamic flowchart self-tuned itself so as to bring the user closer to the solution of the computer problem. As future work, further investigation is needed to optimize the length of rare paths. Also, DFPA could be further tested with different sizes of diagnostic flowcharts, with varying depth and number of decision nodes, in addition to varying threshold values to cater for rare paths.

VIII. References

[1] D. Waterman, A Guide to Expert Systems, Addison-Wesley Publishing Company, 1986, p. 390.

[2] C. Yuan, N. Lao, J. R. Wen, J. Li, Z. Zhang, Y. M. Wang and W. Y. Ma, "Automated known problem diagnosis with event traces," *ACM SIGOPS Operating Systems Review,* vol. 40, no. 4, pp. 375-388, 2006.

[3] A. Beygelzimer, M. Brodie, J. Lenchner and I. Rish, "Automated Knowledge Elicitation and Flowchart Optimization for Problem Diagnosis," in *4th Bayesian Modeling Applications Workshop, UAI (Uncertainty in Artificial Intelligence)*, 2006.

[4] D. Kasper, A. Fauci, S. Hauser, D. Longo, J. Jameson and J. Loscalzo, Harrison's Principles of Internal Medicine 19/E, McGraw Hill Professional, 2015.

[5] J. M. Lee, "A Diagnostic Database System for PC Maintenance and Repair," *Journal of Digital Contents Society,* vol. 9, no. 4, pp. 717-723, 2008.

[6] P. Chen, T. Toyota and Y. Sasaki, "Fuzzy diagnosis and fuzzy navigation for plant inspection and diagnosis robot," in *Proceedings of 1995 IEEE International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium*, 1995.

[7] Oxford Advanced Learner's Dictionary, "Flow Chart," 2016. [Online]. Available: http://www.oxforddictionaries.com/definition/learner/flow-chart. [Accessed 15 Mar 2016].

[8] J. S. Vitter, "Design and analysis of dynamic Huffman codes," *Journal of the ACM (JACM),* vol. 34, no. 4, pp. 825-845, 1987.

[9] R. Solingen, V. Basili, G. Caldiera and H. D. Rombach, "Goal question metric (gqm) approach," *Encyclopedia of Software Engineering,* 2002.

[10] N. K. Denzin and Y. S. Lincoln, Qualitative research, Denzin: NK y Lincoln YS, 2005.

[11] FixingMyComputer, "Computer Repair," FixingMyComputer, 2013. [Online]. Available: http://fixingmycomputer.com/. [Accessed 25 May 2015].

[12] W. Szwoch, "Recognition, understanding and aestheticization of freehand drawing flowcharts," in *IEEE Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007.