QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

WEB TEST CASE GENERATION WITH TIME OPTIMIZATION USING GRAPH DECOMPOSITION

Lana Kamal Mohammed
Department of Software and Informatics Engineering, College of Engineering, Salahaddin University, Erbil, Kurdistan Region, Iraq
Email: lana.kamal.m.a@gmail.com

Moayad Y. Potrus
Department of Information Technology, College of Engineering and Computer Science, Lebanese French University, Kurdistan Region, Iraq
Email: moayad.yousif@lfu.edu.krd

ARTICLE INFO

Article History:
Received:1/7/2023
Accepted: 5/9/2023
Published: Winter2024

Keywords:
Graph Theory, web application testing (WAT), dynamic website, test case generation, software testing, Automated testing, Automated model-based testing

Doi:
10.25212/lfu.qzj.9.4.54

ABSTRACT

Web application testing is an essential process in software development that has become increasingly important in recent years because of the growing complexity and dynamic nature of web applications, in addition to the need to ensure their quality in a highly competitive market. The dynamic nature of web applications is one among the major difficulties in the field of testing web applications. Also, it is crucial to strike a balance between time to market and software quality. Furthermore, in an era of rapid technological advancement, web application testing remains an ongoing and evolving challenge for developers. Thus, this research proposes an approach to optimize the time of the testing phase in the software development life cycle. Within this work, the web application is represented as a graph model, which is then decomposed to facilitate the automatic generation of test cases. And presents an updated graph decomposition algorithm and demonstrates its usefulness in partitioning the graph models of web applications under test. Based on the results, applying the suggested work can reduce the testing time by ~90%. This significant time-saving potential not only accelerates the development lifecycle but also enables developers to allocate more resources to comprehensive testing, leading to more robust and reliable

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

web applications.

1. Introduction

Software testing is categorized as a risky phase in the Software development life cycle (Jamil et al., 2016). With the huge increase of internet, content and the flows of internet infrastructure, the testing becomes more challenging, especially for dynamic websites which are more complex than other types of software due to the change of their content based on users interests, time zones, user's spoken language and many more parameters (Wen, 2001). Consequently, optimizing time spent in the testing phase and increasing accuracy is strongly needed (Yahaya et al., 2017).

Studies have shown that a large number of websites on the internet does not meet the quality standards (Yahaya et al., 2017). In websites, links can break for many reasons. Some links may be changed or renamed or removed which causes the link to break. Therefore, it is important to verify all of the connections on the website's pages again (Khan et al., 2019). Depending on the size of the website and the number of links on each page, it might take a very long period to check every link on every page. The very worst situation for manually checking website links is that each link must be clicked individually in order to report any broken links by determining whether or not the given link is loading.

Software automation testing can speed up and simplify the testing process (Fewster & Graham, 1999). As a result, automating the procedure of checking website links provides an easy solution for ensuring the quality of the software prior to the production phase (Miller, 2005). Since the servers and data used in the development and production environments are distinct from one another. In accordance with that, it is advantageous to check the links before launching the website in order to identify any damaged links in the production environment.

It takes a while to verify all the web connections consecutively, even with automated tools. All of a website's connections can be visualized as a graph with each edge representing a link. Graph Decomposition is the process of dividing a

graph into sub graphs where each edge in the graph corresponds to precisely one of the sub graphs (Arumugam et al., 2013). Thus, by using one of the graph decomposition methods to partition the website link graph into smaller communities, the process of validating the links will go more quickly.

As mentioned in (Setiani et al., 2019) review, there isn't much research about software testing in web-based applications. Thus, In order to increase the quality of websites, especially dynamic websites, this paper aims to solve one of the deployment issues which are the problem of broken links in the production environment. In this research work, test cases are automatically generated to help testers determine how to test their software most effectively while using the least amount of time feasible. The test scenarios are built upon the link graph of the website, which forms the basis for their development. Afterward, divide this graph into equal-sized modules. The path coverage approach is utilized to generate test cases for each level of the partitioned graph. As a result, the calculated times for each stage offer the most accurate testing times.

The paper is divided into the following sections: Section 2 provides a background of web testing. Then, the related studies are discussed in Section 3. The methodology is explained in section 4. Finally, the last section presents each and every result found. Finally, section 6 concludes the research.

2. Background

Early websites were straightforward and contained a few static pages that were simple to test and ensure the quality of. However, websites are transforming into more complicated forms of dynamic pages that are altered based on a variety of factors, such as the user, time zone, spoken language and many others. The content and presentation of a static website are combined, whereas a dynamic website's content (data) is kept in a database and its presentation is built automatically using a template. Since the data and the way, the content is presented can be separated on dynamic websites, they have an advantage over static websites. This helps to the website's higher quality and maintainability (Ricca & Tonella, 2003).

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

In software marketing, software quality is going to become more important. The importance of testing procedures will increase over time as this situation changes (Kassab et al., 2017). In a software project, testing might be the most costly phase. According to one of the estimations, the projects testing process consumed more than half of its resources (Kit & Finzi, 1995). If a software bug wasn't discovered sooner, it will cost more to repair. Therefore, the cost of poor-quality software will increase exponentially. Thus, both for software users and developers, increasing software quality and the efficiency of testing phase are considered practical ways to minimize software expenses in the long term (Kasurinen et al., 2010).

Testing outcomes are significantly improved and the overall number of defects is reduced when a tester is aware of the sort of testing that is required (Freeman, 2002). There are many different types of testing that are widely used including structure and behavioral testing. In structure testing, the internal code of the system is relied on to test the system. However, the behavioral testing concentrates on the behavior (functionality) to check if it behaves as expected which is determined on the requirements.

There are many factors that contribute in the development of low-quality software. The complexity of software deployment in distributed and heterogeneous environments, the necessity of customer-centric customization and configuration during deployment, the difficulties of updating and adapting installed software, and the process of un-deployment or deinstallation are just a few of the deployment-related issues (Dearle, 2007). Dearle also emphasizes the current trends, problems, and difficulties in deployment systems, including the requirement for metadata repositories, the emergence of model-driven development tools, and the deployment issues in the wireless sensor-net domains and mobile areas. By resolving deployment issues, program quality can be improved.

Some methods can be used to enhance the quality of software. Graph partitioning is a method used in computer software to minimize a system's complexity and emphasize its components. It entails dividing a program graph, which is a graph structural model of a program displaying the flow relation or link among the program's components (statements), into distinct groups of intervals or segments.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

The reduced graph that results only contains the paths and the intervals between them. This reduced graph can reflect the basic structure of the program graph, making it easier to analyze and understand (Paige, 1977). It is a fundamental tool in the study of software engineering and has many uses, such as computation analysis and structural testing of programs.

3. Related Work

In recent years, the significance of web applications has increased (Zou et al., 2014). Due to this, the field of web application testing is gaining more attention from researchers. In this section, related research about web application testing techniques and tools are discussed. Each research is conducted to cover different subjects in web testing and use different testing methodologies. One of the testing Methodologies that is widely used in software testing called Model-Based Testing (MBT) which involves developing a model of the software under test, and then using that model to generate test cases (Achkar, 2010).

Many researchers used the MBT approach including (Tanida et al., 2013) proposed a technique that automatically tests dynamic websites. The technique dynamically generates the model based on the crawling then uses the generated model to check the website navigations. Although the author (Tanida et al., 2013) is offering higher coverage and a higher level of automation than the other methods but it has some limitations including 1. The reproducibility of counterexamples may depend on the degree of abstraction used in the approach. 2. False positives may occur because the STG calculated by the crawler overestimates the true possibility of the website's set of traces. 3. The implementation of the software tools utilized may affect the evaluation's internal validity.

Another research based on model-based testing (MBT), (Potrus, 2020) developed an algorithm and a strategy for automated testing of websites, which can explore and detect nodes and events of the website under testing, generate a full-model automatically during the exploration process, and break it into smaller sub-models for more accurate and less time-consuming test case generation. Also, (Panthi & Mohapatra, 2017) proposed a MBT approach for dynamic websites. The research

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

gives better results in the process of executing search engines and how various websites work together. Also, it may improve the quality of web applications by covering every transaction and node in the website, which covers the program code indirectly. But the approach may not be suitable and optimal for all web development languages and frameworks.

Some studies applied model-based testing with an UML model containing (Nagowah & Kora-Ramiah, 2017) proposed and evaluated a tool called CRaTCP (Combinatorial Regression and Test Case Prioritization). CRaTCP is a GUI testing approach that offers test case extraction and GUI ripping. The tool utilized for testing a web page with every combination of possibilities of the user interface elements. The proposed work has some limitations, but the most important to mention is that CRaTCP may not be able to handle dynamic web pages that change frequently. This makes the tool less beneficial nowadays since most of the websites feature dynamic pages that change based on a variety of factors.

Another approach is using (Virtual DOM) tree that was used by (Zou et al., 2014) to execute the model on the client and server side, and developing test generation techniques that combine concrete and symbolic executions to expose faults in web applications. The work also involves the use of static analysis to construct V-DOM trees and the development of crawlability measurements that are used to quantify web application characteristics that have an impact on crawling.

In a research, (de Moura et al., 2017) proposed a web application testing technique that is mainly focused on functional testing to help in the conversion of the BPMN models to create test cases for functional automated testing of websites with BPMS support. The approach involves analyzing BPMN-formatted XML files that are from BPMS tools to determine the particular stages or flows that require testing. It creates a table listing all the potential flows that can occur. The table's outcomes are then used in a specialized tool that produces the initial test script code. These scripts are used in the Cucumber and Selenium. Because files containing the scenarios and steps can be cumbersome and creating test scripts can be hard, using this recommended work may be challenging.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

In the latest studies, notable approaches have been introduced for enhancing regression testing in the context of web applications. The researches by Yandrapally & Mesbah (2023) and Nass et al. (2023) introduce innovative approaches that enhance different facets of web application testing and automation. (Yandrapally & Mesbah, 2023) introduced FRAGGEN, a technique that generates tests for web applications using specially designed models. This approach utilizes state abstraction through fragments and analyzes page fragments in detail to improve the accuracy of model inference and test generation. FragGen has some limitations, such as the need for intricate semantic inference beyond basic characteristics, assumptions regarding changes induced by crawlers in live web applications, and constrained generalizability due to controlled assessments. Furthermore, the manual labeling introduces concerns about internal validity, while FragGen's suitability is primarily confined to web applications and doesn't directly extend to broader domains.

In contrast, (Nass et al., 2023) introduced Similo, a similarity-based technique for locating web elements in test automation. Similo computes similarity scores between locator parameters to find similar web elements, leveraging attributes, positions, texts, and images for repair suggestions. The Java-based implementation seamlessly integrates with Selenium test suites. Similo approach has some limitations including assuming correct location based on a single locator and a failure rate of 11% in its experimental study, contrasting with 27% for a theoretical LML variant. Also, its effectiveness varies with changes' complexity.

Test cases can be created by using the techniques for gathering interaction of the user in pairs of names and values and URL form. Such a technique called Leveraging User Session Data as used in some research (Elbaum et al., 2005) (Sampath & Bryce, 2012). (Elbaum et al., 2005) presented a testing approach for Web-based applications which leverages captured data during users' sessions to create test cases. (Elbaum et al., 2005) has some limitations including 1. Not all Web apps may be represented by this approach. 2. May not capture all possible user interactions or edge cases. 3. The effectiveness may be limited by no. of users' sessions collected and cost spend in collecting and analyzing additional data. 4. May not be effective

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

for detecting faults that are not likely to be encountered during normal user operations, such as faults that require specific name-value pairs.

(Sampath & Bryce, 2012) proposed a method to improve the rate at which faults are detected based on user sessions by ordering reduced test suites. The authors offer multiple methods for ordering test suites based on the priority criteria. They also developed Mod_APFD_C which is a measurement for comparison between the various sized test suites and it includes the cost. The main limitations of (Sampath & Bryce, 2012) is that the effectiveness of the approach may depend on the characteristics of web apps being tested and quality of the usage logs used to generate test cases. And, it may require significant computational resources to generate and order the reduced test suites, which may not be feasible for large web applications with huge no. user sessions.

Also, Reverse engineering approach can be useful for testing web apps in several ways. One of the main benefits is that it allows testers to generate models of an existing web app through the execution, which can be used to understand the application's design, architecture, and functionality. The generated model can then be used in the test case creation process and automate the testing process, which can save time and improve the accuracy of the testing. Additionally, reverse engineering can help identify anomalous behaviors in the application, which can be used to improve the application's quality and performance. Finally, reverse engineering can be used to analyze the navigation habits of users, which can help improve the user experience of the application.

In (Draheim et al., 2005) described a tool called Revangie for form-oriented analysis, which reconstructs analytical models using black box reverse engineering. This research work also used clustering methods and statistical testing to refine and coarsen the model as needed. The main limitation of (Draheim et al., 2005) is that initial models generated by the proposed tool may not be optimal for all use cases.

Some other techniques used by (Artzi et al., 2010) called explicit state model checking which they used for finding faults in web apps that used PHP by using dynamic test generation, combined concrete and symbolic execution. The technique is implemented in a tool called Apollo, which automatically generates tests and can

be used for finding and fix underlying faults. There have been several recent additions to the field of web-based application testing, as an example the proposal of a new coverage criterion called hybrid coverage for dynamic web testing proposed by (Zou et al., 2013) which combines statement coverage and HTML element coverage to cover both client-side and server-side features. The approach uses a web crawler to generate a web UI model and collects test data at runtime to calculate coverage information. According to the findings, hybrid coverage is more effective at finding bugs than element coverage and statement coverage.

Some studies offered structural testing as a technique for their work, including (Artzi et al., 2010), while others utilized behavioral testing, including (Sampath & Bryce, 2012) (Potrus, 2020). (Elbaum et al., 2005) (Panthi & Mohapatra, 2017) used functional testing which is a type of behavioral testing. Also, (Yandrapally & Mesbah, 2023) and (Nass et al., 2023) mainly focused on regression testing. Some other studies used both structural and behavioral testing (Zou et al., 2013) (Zou et al., 2014).

Although certain studies' research is utilized to test web-based apps (Elbaum et al., 2005) (Artzi et al., 2010) (Sampath & Bryce, 2012) (Zou et al., 2014) (de Moura et al., 2017) (Yandrapally & Mesbah, 2023) and (Nass et al., 2023). However, other research focuses specifically on dynamic web applications (Zou et al., 2013) (Panthi & Mohapatra, 2017).

While (Zou et al., 2013) (Elbaum et al., 2005) (Panthi & Mohapatra, 2017) and other proposed works are semi-automatic, some are fully automated and don't require any human interaction like (Artzi et al., 2010) (Yandrapally & Mesbah, 2023) and (Nass et al., 2023).

Such tools that are used in developing the proposed testing approaches including Crawljax, Selenium Webdriver, Xdebug and PHPUnit, PHP, Schoolmate and Timeclock, AJAX, ChoiceFinder, VeriSoft, WebNavigator, and Error Checker, Microsoft JET Database Engine, Crawlers such as LinkCheck, SiteInspector, Weblint, Webtrends, HTML verifier, NuSMV, WebKing, Cucumber, Apache POI and JUnit. Among all of these tools, Selenium Webdriver and Crawljax are the most commonly used.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Previous research has explored diverse testing techniques and methodologies for web applications, including Model-Based Testing (MBT), dynamic analysis, reverse engineering, and various coverage criteria. Table 1 provides a comprehensive comparison of the related work. Researchers have aimed to enhance the efficiency and effectiveness of web app testing, addressing dynamic content, coverage, and scalability. While this study focuses on optimizing web app testing through a Model-based approach to tackle challenges from dynamic web nature to comprehensive testing with reduced time. This research introduces a graph decomposition algorithm inspired by Karger's, tailored for web app testing.

Additionally, this study emphasizes multithreading for parallel test case execution, resulting in faster testing. Performance is evaluated across different partitioning levels for an optimal balance. Visualization techniques offer insights into web app structure, aiding understanding and testing. By decomposing the web app graph and generating subgraph-specific test cases, this approach significantly reduces testing time while ensuring thorough coverage. It effectively addresses dynamic web app challenges and shows promising results in time reduction and accuracy enhancement.

Table (1): comprehensive comparison of the related work

Reference	Techniques used	Test Strategy	Application	Automation level	Tools, technologies used	Main Limitation
(Elbaum, 2005)	user-session-based testing	functional testing, behavioral testing	Web applications	Semi-automated	-	Could overlook certain interactions or edge cases, Effectiveness limited by session quantity and costs, May not catch rare faults or specific issues, Requires extra effort to set up and maintain data infrastructure.
(Draheim, Dirk, 2005)	Reverse engineering, statistical testing,	behavioral testing	dynamic web sites	Semi-automated	HTTP Client	Initial models from the tool might not suit all cases optimally, The tool aims to restore form-based models of web apps, which might not fit all

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

	and clustering					app types or analysis needs, Accuracy and completeness of the Revangie model could be impacted by aspects like app complexity, dynamic content, and client-side scripting.
(Achkar, 2010)	Model Based Testing (MBT)	behavioral testing	Web applications	Semi-automated	TestOptimal tool	intricacy of creating and maintaining accurate models, potential incomplete coverage, reliance on specific tools, necessary human involvement, and challenges in scaling for complex and large applications.
(Artzi, 2010)	explicit-state model checking	Structural testing	PHP web applications	Fully automated	HTML verifier	Limited sources of input parameters, Limited forms of constraints to be solved, Simulating user inputs based on locally executed JavaScript, Limited external validity.
(Sampath, 2012)	User-session-based testing, test suite reduction, test suite prioritization, concept-analysis-based reduction technique	behavioral testing	Web applications	Semi-automated	replay tool, CPUT (Combinatorial-based Prioritization for User-session-based Testing)	The approach's success relies on the specific attributes of the tested web application and the quality of the usage logs employed to create test cases, Creating and organizing streamlined test suites could necessitate considerable computational power, which could be impractical for sizable web applications with numerous user sessions.
(Tanida 2013)	Model based testing (MBT)	behavioral testing	dynamic web applications	Semi-automated	Dynamic crawling, Model checking techniques, NuSMV,	Counterexample reproducibility relies on the approach's abstraction level, Evaluation's internal validity hinges on software tool implementation, Model generation is intrinsically incomplete due to resource

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

					AJAX, Selenium, WebKing, and Sahi, Template-based specification language, Crawljax.	constraints, Crawling time limits large-state web apps, Crawler's computed STG is an over-approximation, leading to potential false positives.
(Zou, Yunxiao, et al. 2013)	statement coverage and HTML element coverage, UI model, Test case prioritization	both structural and behavioral testing	dynamic web applications	Semi-automated	Crawljax, Selenium, Xdebug and PHPUnit, PHP, Schoolmate and Timeclock, AJAX, HTML	Limited number of subject programs, incomplete web UI model, limited coverage criteria, limited fault-detect rate, limited applicability, limited scalability.
(Zou, 2014)	Virtual DOM coverage, model based testing	combination of both structural and behavioral testing	web applications	Semi-automated	Apollo, Crawljax, Web crawlers, AJAX	Incomplete client-side code coverage, Challenge in measuring test sufficiency, Limited dynamic web app feature handling, Reliance on manual inputs introducing human error and increasing testing effort, Limited scalability.
(Panthi, 2017)	model-based testing	behavioral testing, functional testing	Dynamic web application	Semi-automated	JTSG algorithm	may not be suitable for all web development languages and frameworks. It may not be optimal for all web applications and may require further optimization techniques.
(Nagowah, 2017)	Modelling Language (UML), Model-Based Testing	GUI testing and event-driven testing	web based applications	Semi-automated	Selenium WebDriver, Apache POI, JUnit, Eclipse.	Offers limited HTML element support, focusing on single-page analysis, lacking whole-website examination. Validation rules solely obtainable from JavaScript. Relies on coding standards for proper system

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

						function. Inadequate for complex apps with multiple controls and validations, struggles with rapidly changing dynamics. Falters with intricate user interactions, such as drag-and-drop, pop-ups, and multi-tab interfaces.
(de Moura, 2017)	BPMN (Business Process Model and Notation) model, which is a form of model-based testing	functional testing	Web applications	Semi-automated	BPMN, BPMS, XML, Selenium, Cucumber, programming languages such as Java and Python.	The files containing the scenarios and steps can be cumbersome, Creating test scripts can also be challenging, Technical limitations such as improving the treatment of processes with recursion and recognition of the different names of tags.
(Potrus, Moayad Y. 2020)	Model-Based Testing (MBT)	Behavioral testing	Web applications	fully automated	Google Chrome extension	Effectiveness hinges on accurate model construction reflecting expected behavior, Limited to black-box testing, omitting internal code structure, possibly missing certain errors, Suitability varies, complex websites may not model accurately, Resource demands might hinder scalability, particularly for large sites.
(Yandrapally & Mesbah, 2023)	Model-Based Testing (MBT), Fragment-based Analysis	Regression testing	Web applications	fully automated	WebDriver API in Java, VIPS Algorithm, APTED and Histogram, CRAWLJAX Plugin	Semantic Inference, Test Oracle Generation, Limited Generalizability, Manual Labeling, Limited Scope

(Nass et al., 2023)	Levenshtein Distance, Comparison of Locator Parameters, Integration with Java-based Selenium Test Suites, Weighted Approach	regression testing at the graphical user interface (GUI) level	web applications	fully automated	java programming language, selenium webdriver	Reduced sensitivity to technical issues, Threshold determination challenge, Limited evaluation scope, Selection of locator parameters, Lack of optimization in weights

4. Methodology

In this research work, duration time to check the broken links were calculated and tried to find the optimal. The procedure involved visualizing website links through site graphs using the "requests" and "Beautiful Soup" libraries. Then graph decomposition technique was utilized to partition the graph model generated in the first step using the Karger algorithm, with modifications for balanced graph partitioning. Test cases were generated through path coverage, ensuring comprehensive edge coverage. During the execution of these test cases, the implementation of multithreading through Python's threading module enabled parallelized execution.

Several example websites with more than 100 pages each were used, providing substantial data for analysis. Due to simplicity, readability, and flexibility, Python programming language was used, which is a widely used high-level, interpreted computer language for a variety of applications such as web developing, artificial intelligence, data analysis, and scientific computing (Yuill & Halpin, 2006).

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

4.1 Web Graph Visualization

To visualize the website links, a site graph is used that was presented by (Tomlinson, 2020) The program written in python which automatically lists all the links of an example website in a form of graph that uses it to create test cases in the next steps. Each node in the graph corresponds to a page on the website, and each edge represents the link that redirects the user from one page to another. The site graph is to fetch the web pages, “requests” library was used. And to parse each webpage, “Beautiful Soap” was used. pyvis which uses vis.js was used to draw and visualize the graph. An example website is presented in figure 1.

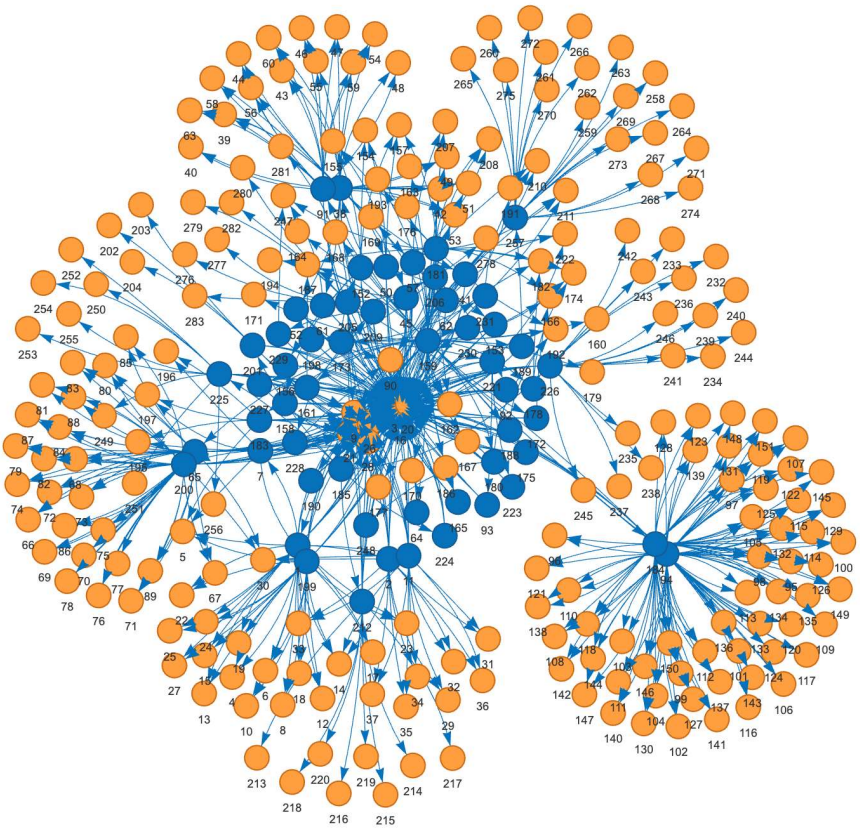


Figure (1): Website graph visualization

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

4.2 Web Graph Decomposition

There are many graph decomposition algorithms to use to partition a graph but for the proposed work, the algorithm should have some specific properties to fit the research needs. First, the algorithm should be applicable for directed graphs because the links in the website redirect a page (starting node) to another page (destination node). Second, the algorithm used should divide the graph into 2n sub graphs where n denotes the number of nodes. And each sub-graph should have the same number of nodes. Third, the algorithm should divide the graph with min-cut (minimum cut) which is the smallest number of edges required to divide the graph into two disjoint subgraphs.

A technique created by Michelle Girvan and Mark Newman in 2002 for identifying community structure in networks is known as Girvan and Newman which looks for community edges using edge betweenness information. The Girvan-Newman technique becomes impractical for very large graphs, because of its scalability. The algorithm's running time is proportional to the graph's number of edges, which can be computationally expensive for large graphs which make it impossible to identify communities in huge networks (Girvan & Newman, 2002). Because of this, this method was not used.

Brian Kernighan and Shen Lin introduced a heuristic method known as Kernighan-Lin for partitioning graphs. When dividing a graph's nodes into groups of a certain size, the algorithm tries to reduce the total cost of all connections cut. (Kernighan & Lin, 1970). But there are some drawbacks to the Kernighan-Lin method, including: 1. The algorithm is not practical for large graphs because it has O(n3) time complexity, where n denotes the number of graph vertices. The algorithm might take a while to process big graphs. 2. For graphs with weighted or directed edges, it might not be suitable. 3. The algorithm may not perform well in graphs with highly unequal node sizes. It might fail to identify a suitable split.

Another community detection method that presented by (Blondel et al., 2008) can be used for large networks called Louvain algorithm which utilizes modularity optimization. Several drawbacks of the Louvain algorithm are: 1. the algorithm may not be capable of identifying communities on a smaller scale than a particular size,

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

which is a known limitation of modularity optimization. 2. The accuracy of the algorithm is not as good as some other algorithms. 3. The algorithm's output is determined by the sequence that each of the nodes are analyzed, which can influence the computation time.

Another partitioning algorithm that uses the matrix's eigenvectors calculated from point distance called spectral partitioning (Ng et al., 2001). The spectral partitioning algorithm has some limitations including: 1. Many algorithms that use eigenvectors have no proof that they will actually compute a reasonable clustering. 2. Spectral partitioning methods tend to focus on simplified algorithms that only use one eigenvector at a time, which may not be optimal. 3. Spectral partitioning is computationally expensive for large datasets. 4. Spectral partitioning may not work well for datasets with complex structures or noise.

Among different decomposition algorithms, the Karger algorithm can be used which is a randomized method for searching undirected weighted graphs for global min-cuts. The technique is extremely straightforward, and it works by continuously contracting randomly selected edges until there are only two nodes remaining which are the two sides of the min-cut (Karger, 1993). Because of the randomization that Karger uses, the algorithm might not find the exact min-cut, the estimate's accuracy is influenced by the graph's size and the number of repetitions. Based on that, as the number of iterations rises, so does the probability of finding the right min-cut.

Karger's algorithm has the following advantages: 1. It is easy and simple to use. 2. It is really effective and efficient due to the use of randomization to determine the graph's min-cut. 3. It can efficiently handle large-scale graphs containing a lot of nodes and edges because of the polynomial time complexity of the algorithm. 4. Despite being randomized, the algorithm has a high probability of discovering the accurate min-cut. 5. The method can be modified easily to locate approximative min-cuts or to resolve associated issues, like the max-flow issue.
Despite having all of these advantages, Karger has the issue of not partitioning the graph into equal sized communities. To adjust the Karger algorithm a new version of the algorithm is presented. The steps of the Modified Karger Algorithm are shown

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

below. The modified version contracts the randomly selected node with its neighbors until half nodes in the graph are contracted which represent one community and the rest of the nodes be the other community.

let n be the total no. of nodes in the original graph
select a random node x
repeat until no. of nodes equals (n/2 + 1)
    contract x with a neighbor node

The presented work decomposes the graph into equal sized subgraphs repeatedly by 2n, where n denotes the number of iterations. The decomposition stops when one of the communities has only one node which cannot be divided more. To cover the cut-edges, they are added to the community where the starting node is located. Sample graph decomposition is presented in figure 2.

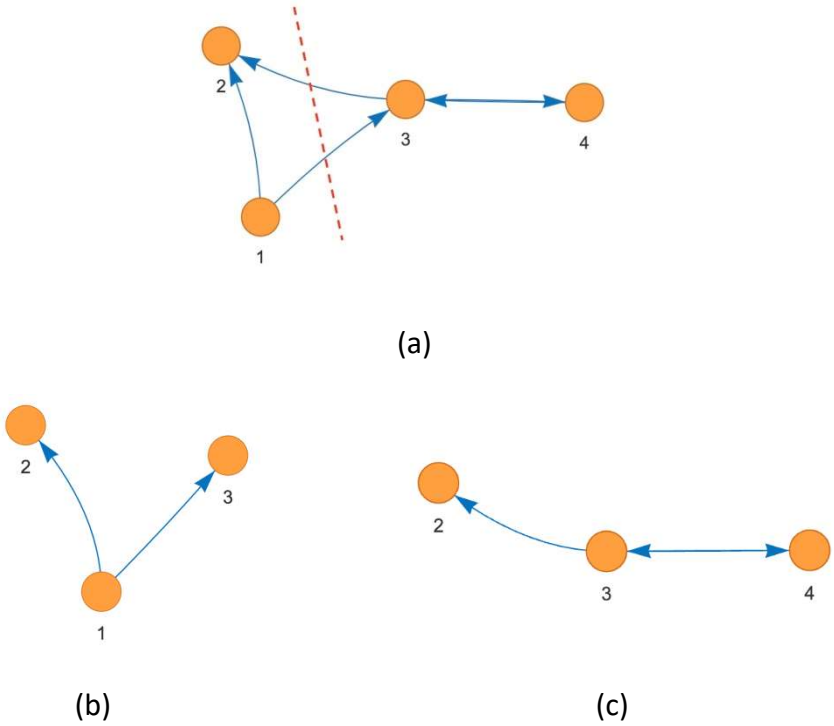


Figure (2): Sample graph (a) decomposed to subgraphs shown in (b) and (c)

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

4.3 Test Case Generation

The path coverage technique was used to create the test cases to ensure the covering of all the possible paths. Each test case covers a list of nodes and edges which represents a link in the website. A graph with directed edges, also known as arcs, connecting its nodes is known as a directed graph (or digraph) (Bang-Jensen & Gutin, 2008). For a node x in a directed graph, an edge that connects node x as starting node (origin node) to another node is called outgoing edge or out-edge of node x. An edge that connects a node-to-node x as an end node (destination node) is called incoming edges or in-edges of node x. Generated test cases are covering all the edges in the graph. For creating test cases, the following technique is used:

```
Repeat until all edges are covered
        Create a new test case with path p
        Select a random edge x and add to p
                While x has an in-edge
                        Connect x with an in-edge and add to p
                While x has an out-edge
                        Connect x with an out-edge and add to p
```

4.4 Multithreading

Multithreading is a model for parallel computing in which several CPU cores work on a single set of input data. It is possible to consider tasks distributed across several threads to be sub-tasks that are accessing the same memory locations (Khot, 2017). Multithreading allows for parallel processing of tasks, which can lead to improved performance and faster execution times. In the proposed work, multithreading was utilized to speed up and reduce time spent during the process of executing the test cases. In Python programming language which is used in this research, multithreading can be achieved using the `threading` module. In this work, each partition level uses the same number of threads as subgraphs at that level. And each thread is generated for each subgraph to run the test cases for that specific subparagraph.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

5. Experimental Results and Discussion

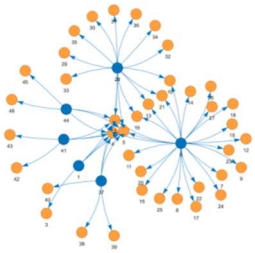
In this proposed work, the following research questions were answered:

- Is the time needed to test website links decreased when it's partitioned into smaller sections?
- What is the optimal partition level for testing an example website based on time optimization?
- Is the graph decomposition technique used in the proposed work efficient to use for testing?

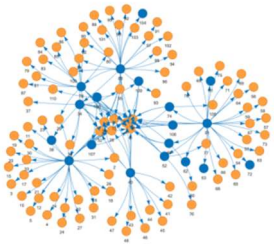
The partitioned website graph into subgraphs is shown in figures 3. The proposed decomposition algorithm was presented based on Karger's algorithm which is more effective and suitable with research needs. Based on the partitioned graph, the subgraphs can be used in the case of dynamic graphs when a part of the website is added or changed or removed. That way, only the subgraphs containing the modification will be affected and should be retested.

(a)

(b)

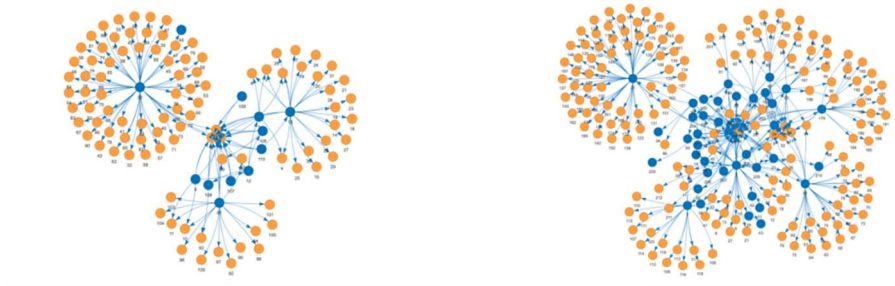


(c)



(d)

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)



(e) (f)

Figure (1): The original web application graph model shown in (figure 1) decomposed to 2 subgraphs (a) and (b). Then, subgraph (a) is decomposed to another 2 smaller subgraphs (c) and (d). And subgraph (b) decomposed to (e) and (f).

In order to find the least possible time to run the tests of a website, some example websites with more than 100 pages each were used which are listed in table 2. The program will create test cases and check the validity of links as displayed in figures 4 and. Test cases are displayed as a path starting from a node (page) to other nodes sequentially.

Table (2): List of case studies for applying the proposed algorithm and their size

#	Website Link	Total no. of pages	Total no. of links
1	https://www.cs.cornell.edu/~kt/	298	730
2	https://cpanel.net/	341	2864
3	https://wiki.python.org/moin/BeginnersGuide/	261	463
4	https://safety.google/	891	7239

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

```
testcase 1: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/', 'https://
www.cs.cornell.edu/~kt/post/', 'https://www.cs.cornell.edu/~kt/post/quine/', 'https://
www.cs.cornell.edu/home/kleinber/']
testcase 2: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/authors/austin-r.-
benson/', 'https://cis.cornell.edu']
testcase 3: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/post/6502-2/',
'https://en.wikipedia.org/wiki/WDC_65C22']
testcase 4: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/authors/kiran-
tomlinson/', 'https://www.cs.cornell.edu/~kt/publication/2018-tomlinson-examining-phylogeny-
inference/', 'https://sourcethemes.com/academic/']
testcase 5: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/guitar', 'https://
www.cs.cornell.edu/~kt/post/guitar-build/', 'https://www.electricherald.com/fender-stratocaster-
templates/']
testcase 6: ['https://www.cs.cornell.edu/~kt/guitar', 'https://youtu.be/LR-T2qTLF6o']
testcase 7: ['https://www.cs.cornell.edu/~kt/', 'https://scl.cornell.edu/coe/lindseth-climbing-
center']
testcase 8: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/graph/', 'https://
www.cs.cornell.edu/~kt/post/site-graph/', 'https://pyvis.readthedocs.io/en/latest/']
testcase 9: ['https://www.cs.cornell.edu/~kt/', 'https://www.cs.cornell.edu/~kt/news', 'https://
www.cs.cornell.edu/~kt/publication/2020-commins-diverging-string-sequences', 'https://github.com/
tomlinsonk/diverging-string-seqs']
testcase 10: ['https://www.cs.cornell.edu/~kt/', 'https://mengtingwan.github.io/']
testcase 11: ['https://www.cs.cornell.edu/~kt/post/6502-2/', 'https://www.cs.cornell.edu/~kt/post/
6502-3', 'https://www.cs.cornell.edu/~kt/post/6502-4', 'https://eater.net/6502']
testcase 12: ['https://www.cs.cornell.edu/~kt/post/site-graph/', 'https://developer.mozilla.org/en-
US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types']
testcase 13: ['https://www.cs.cornell.edu/~kt/authors/kiran-tomlinson/', 'https://www.cs.cornell.edu/
~kt/publication/2019-tomlinson-cyclic-cellular-automaton/', 'https://gohugo.io']
testcase 14: ['https://www.cs.cornell.edu/~kt/post/quine/', 'https://github.com/tomlinsonk']
testcase 15: ['https://www.cs.cornell.edu/~kt/', 'https://web.stanford.edu/~jugander/']
```

Figure (4): Test cases of a sample website

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

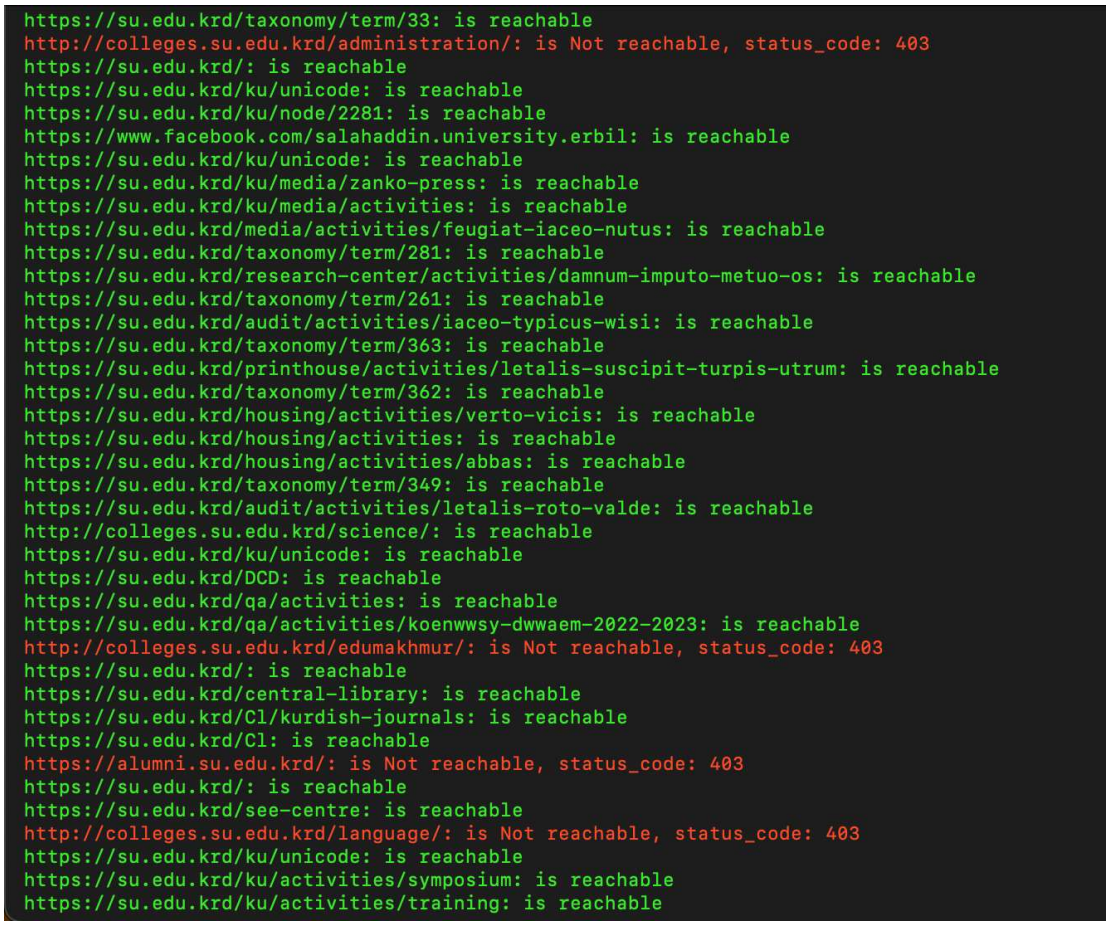


Figure (5): Validity check of website links

Tables 3 and 4 show the results that were discovered. As displayed for each website, the best possible partition is different which is chosen based on the least time duration.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

Table (3): Testing results in example website https://www.cs.cornell.edu/~kt/

#	No. of Test Cases	Time (in second)
whole website	627	1449.747305
partition to 2	647	899.603601
partition to 4	656	722.621033
partition to 8	663	593.991405
partition to 16	668	361.184642
partition to 32	670	250.318771
partition to 64	674	133.38867
partition to 128	675	132.510486
partition to 256	678	140.233086

Table (4): Testing results of example website https://cpanel.net/

#	No. of Test Cases	Time (in second)
whole website	2784	5965.955895
partition to 2	2798	4245.793578
partition to 4	2804	3242.491023
partition to 8	2822	2176.939256
partition to 16	2834	1774.1613
partition to 32	2841	1479.395243
partition to 64	2845	1016.846121
partition to 128	2853	682.450128
partition to 256	2855	635.457315

Based on the results found, as the graph is decomposed, the testing time decreases until it reaches its limit when the number of test cases is high, which affects the time of the test. As shown in figure 6, an example website (https://glassdoorspecialist.com/) is used. It is shown that at first, the time duration of the test is really high. Then it decreases as the website is partitioned more. The number of test cases is always increasing due to the breaking of the graph. However, the time will go from decreasing to increasing, as depicted in figure 7 below.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

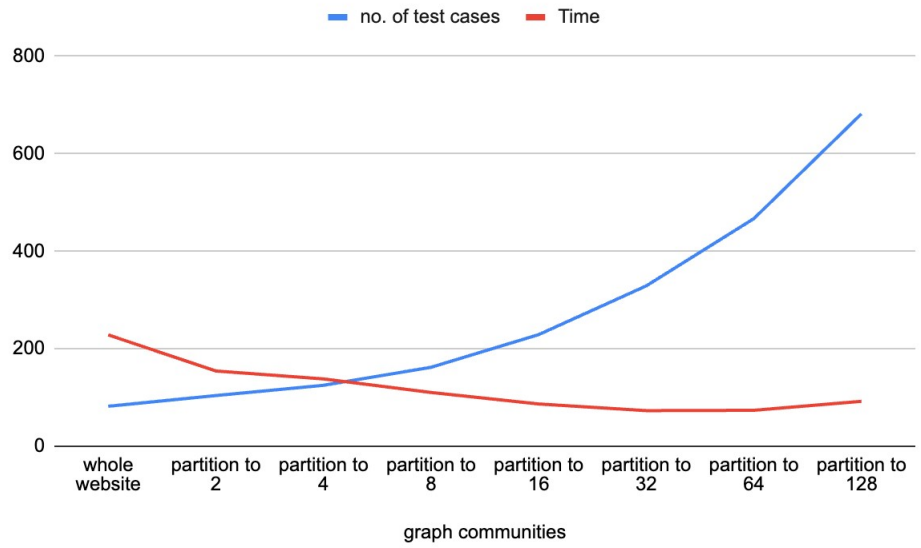


Figure (6): The effect of test case length on testing time

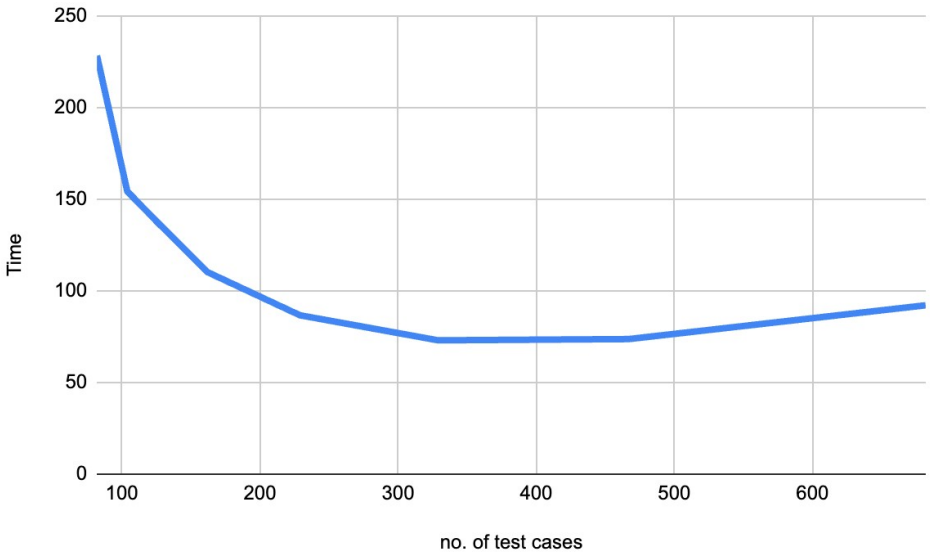


Figure (7): Optimal testing time length based on the number of test cases

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

6. Conclusion

In this research work, an approach is proposed for optimizing time spent in the testing phase of software development lifecycle and increasing accuracy of dynamic web applications.

Graph visualization was automatically created to visualize the website. Also, an Updated decomposition algorithm is presented to decompose the visualized graph into equal sized subgraphs. The graph decomposition approach can be really useful in the case of regression testing, especially for dynamic web applications which are frequently changed based on many factors. After each modification, insertion or deletion of any section of the website, the only sub-sections can be retested which will lead to optimize the time of the testing process. Then, test cases are created and executed using the multi-thread module in python.

According to the results, the time required is reduced when the graph is decomposed until it approaches its maximum, where the total no. of test cases has a direct effect on how long the test takes.

The suggested study can be improved further by applying various forms of testing. Also, it can be integrated with other automated testing tools to achieve better results.

References:

1. *Achkar, H. (2010). Model based testing of web applications. Proceedings of 9th Annual STANZ, Australia.*
2. *Artzi, S., Kiezun, A., Dolby, J., Tip, F., Dig, D., Paradkar, A., & Ernst, M. D. (2010). Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking. IEEE Transactions on Software Engineering, 36(4), 474–494. https://doi.org/10.1109/TSE.2010.31*
3. *Arumugam, S., Hamid, I., & Abraham, V. M. (2013). Decomposition of graphs into paths and cycles. Journal of Discrete Mathematics, 2013.*
4. *Bang-Jensen, J., & Gutin, G. Z. (2008). Digraphs: Theory, algorithms and applications. Springer Science & Business Media.*
5. *Benedikt, M., Freire, J., & Godefroid, P. (2002). VeriWeb: Automatically Testing Dynamic Web Sites.*

6. *Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.*
7. *de Moura, J. L., Charao, A. S., Lima, J. C. D., & de Oliveira Stein, B. (2017). Test case generation from BPMN models for automated testing of Web-based BPM applications. 2017 17th International Conference on Computational Science and Its Applications (ICCSA), 1–7. https://doi.org/10.1109/ICCSA.2017.7999652*
8. *Dearle, A. (2007). Software deployment, past, present and future. Future of Software Engineering (FOSE'07), 269–284.*
9. *Di Lucca, G. A., & Di Penta, M. (2003). Considering browser interaction in Web application testing. Fifth IEEE International Workshop on Web Site Evolution, 2003. Theme: Architecture. Proceedings., 74–81. https://doi.org/10.1109/WSE.2003.1234011*
10. *Draheim, D., Lutteroth, C., & Weber, G. (2005). A Source Code Independent Reverse Engineering Tool for Dynamic Web Sites. Ninth European Conference on Software Maintenance and Reengineering, 168–177. https://doi.org/10.1109/CSMR.2005.4*
11. *Elbaum, S., Rothermel, G., Karre, S., & Fisher II, M. (2005). Leveraging user-session data to support Web application testing. IEEE Transactions on Software Engineering, 31(3), 187–202. https://doi.org/10.1109/TSE.2005.36*
12. *Fewster, M., & Graham, D. (1999). Software test automation. Addison-Wesley Reading.*
13. *Freeman, H. (2002). Software testing. IEEE Instrumentation & Measurement Magazine, 5(3), 48–50.*
14. *Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. Proceedings of the National Academy of Sciences, 99(12), 7821–7826.*
15. *Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. (2016). Software Testing Techniques: A Literature Review. 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M), 177–182. https://doi.org/10.1109/ICT4M.2016.045*
16. *Karger, D. R. (1993). Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. Soda, 93, 21–30.*
17. *Kassab, M., DeFranco, J. F., & Laplante, P. A. (2017). Software testing: The state of the practice. IEEE Software, 34(5), 46–52.*
18. *Kasurinen, J., Taipale, O., & Smolander, K. (2010). Software test automation in practice: Empirical observations. Advances in Software Engineering, 2010.*
19. *Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 49(2), 291–307.*
20. *Khan, F. N., Ali, A., Hussain, I., Sarwar, N., & Rafique, H. (2019). Repairing Broken Links Using Naive Bayes Classifier. In I. S. Bajwa, F. Kamareddine, & A. Costa (Eds.),*

Intelligent Technologies and Applications (pp. 461–472). Springer. https://doi.org/10.1007/978-981-13-6052-7_40

21. *Khot, T. (2017). Parallelization in Python. XRDS: Crossroads, The ACM Magazine for Students, 23(3), 56–58.*
22. *Kit, E., & Finzi, S. (1995). Software testing in the real world: Improving the process. ACM Press/Addison-Wesley Publishing Co.*
23. *Lei Xu, Baowen Xu, Zhenqiang Chen, Jixiang Jiang, & Huowang Chen. (2003). Regression testing for Web applications based on slicing. Proceedings 27th Annual International Computer Software and Applications Conference. COMPAC 2003, 652–656. https://doi.org/10.1109/CMPSAC.2003.1245411*
24. *Miller, E. (2005). Website testing. Companion Paper of "The Web Site Quality Challenge". Proceedings of QW 1998 Conference. Https://Bit. Ly/2WIj1vp [Accessed: 15 May 2020].*
25. *Nagowah, L., & Kora-Ramiah, K. (2017). Automated complete test case coverage for web based applications. 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), 383–390. https://doi.org/10.1109/ICTUS.2017.8286037*
26. *Nass, M., Alégroth, E., Feldt, R., Leotta, M., & Ricca, F. (2023). Similarity-based Web Element Localization for Robust Test Automation. ACM Transactions on Software Engineering and Methodology, 32(3), 1–30. https://doi.org/10.1145/3571855*
27. *Ng, A., Jordan, M., & Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. Advances in Neural Information Processing Systems, 14.*
28. *Paige, M. R. (1977). On partitioning program graphs. IEEE Transactions on Software Engineering, 6, 386–393.*
29. *Panthi, V., & Mohapatra, D. P. (2017). An approach for dynamic web application testing using MBT. International Journal of System Assurance Engineering and Management, 8(S2), 1704–1716. https://doi.org/10.1007/s13198-017-0646-0*
30. *Potrus, M. Y. (2020). GENERATING MODELS OF SOFTWARE SYSTEMS DURING EXPLORATORY. Zanco Journal of Pure and Applied Sciences, 32(4), 12–21.*
31. *Ricca, F., & Tonella, P. (2003). Using clustering to support the migration from static to dynamic web pages. 11th IEEE International Workshop on Program Comprehension, 2003., 207–216.*
32. *Sampath, S., & Bryce, R. C. (2012). Improving the effectiveness of test suite reduction for user-session-based testing of web applications. Information and Software Technology, 54(7), 724–738. https://doi.org/10.1016/j.infsof.2012.01.007*
33. *Setiani, N., Ferdiana, R., Santosa, P. I., & Hartanto, R. (2019). Literature review on test case generation approach. Proceedings of the 2nd International Conference on Software Engineering and Information Management, 91–95.*

34. *Tanida, H., Prasad, M. R., Rajan, S. P., & Fujita, M. (2013). Automated System Testing of Dynamic Web Applications. In M. J. Escalona, J. Cordeiro, & B. Shishkov (Eds.), Software and Data Technologies (Vol. 303, pp. 181–196). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-36177-7_12*
35. *Tomlinson, K. (2020, March 15). Site Graph. Kiran Tomlinson. https://www.cs.cornell.edu/~kt/post/site-graph/*
36. *Tonella, P., & Ricca, F. (2002). Dynamic model extraction and statistical analysis of Web applications. Proceedings. Fourth International Workshop on Web Site Evolution, 43–52. https://doi.org/10.1109/WSE.2002.1134088*
37. *Wen, R. B. (2001). URL-driven automated testing. Proceedings Second Asia-Pacific Conference on Quality Software, 268–272. https://doi.org/10.1109/APAQS.2001.990029*
38. *Yahaya, J. H., Ibrahim, A. A., & Deraman, A. (2017). Software Process Model for Dynamic Website Development towards Quality Product. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9(3–3), Article 3–3.*
39. *Yandrapally, R., & Mesbah, A. (2023). Fragment-Based Test Generation For Web Apps. IEEE Transactions on Software Engineering, 49(3), 1086–1101. https://doi.org/10.1109/TSE.2022.3171295*
40. *Yuill, S., & Halpin, H. (2006). Python. Python Releases Wind, 24.*
41. *Zou, Y., Chen, Z., Zheng, Y., Zhang, X., & Gao, Z. (2014). Virtual DOM coverage for effective testing of dynamic web applications. Proceedings of the 2014 International Symposium on Software Testing and Analysis, 60–70. https://doi.org/10.1145/2610384.2610399*
42. *Zou, Y., Fang, C., Chen, Z., Zhang, X., & Zhao, Z. (2013). A Hybrid Coverage Criterion for Dynamic Web Testing.*

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

دروستکردنی کەیسی تاقیکردنەوە بۆ بەرنامەی وێب وە کەمکردنەوەی کاتی تێچوو بە بەکارهێنانی تەکنیکی دابەشکردنی هێڵکاری

پـوختـه:

پڕۆسەی تاقیکردنەوەی بەرنامەی وێب پرۆسەیەکی بنەڕەتییە لە پەرەپێدانی سۆفتوێر کە لەم ساڵانەی دواییدا گرنگییەکی زیاتری بەخۆیەوە بینیوە بەهۆی زیاتر ئاڵۆزبوون و سروشتی دینامیکی بەرنامەکانی وێب، وە پێویستی زیاتری دڵنیابوون لە کوالیتییان لە خستنە بازاڕی پڕ لە کێبڕکێدا. سروشتی داینامیکی بەرنامەکانی وێب یەکێکە لە کێشە سەرەکییەکانی بواری تاقیکردنەوەی بەرنامەکانی وێب. هەروەها زۆر گرنگە هاوسەنگی بکرێت لە نێوان کاتی خستنە بازاڕ و کوالیتی سۆفتوێرەکە. جگە لەوەش، لە سەردەمی پێشکەوتنی خێرای تەکنەلۆژیادا، تاقیکردنەوەی بەرنامەکانی وێب وەک کێشەیەکی بەردەوام و پەرەسەندوو دەمێنێتەوە بۆ گەشەپێدەران. بۆ ئەم مەبەستە ئەم توێژینەوەیە ڕێگەیەک پێشنیار دەکات بۆ کەمکردنەوەی کاتی تێچوو لە قۆناغی تاقیکردنەوە لە سووڕی ژیانی گەشەپێدانی سۆفتوێر. لەناو ئەم کارەدا، بەرنامەی وێب وەک مۆدێلێکی گراف نیشان دەدرێت، کە دواتر دابەش دەکرێت بۆ ئاسانکاری لە دروستکردنی ئۆتۆماتیکی کەیسیەکانی تاقیکردنەوە. هەروەها ئەلگۆریتمێکی نوێکردنەوەی (Graph Decomposition) پێشکەش دەکرێت و سوودەکەی نیشاندەدرێت لە دابەشکردنی مۆدێلی گرافی بەرنامەکانی وێب کە لەژێر تاقیکردنەوەدان .بە پێی ئەنجامەکان، بەکارهێنانی ئەم کارە پێشنیار کراوە دەتوانێت کاتی تێچووی تاقیکردنەوە بە ڕێژەی نزیکەی ٩٠% کەم بکاتەوە. ئەم توانایە گرنگەی پاشەکەوتکردنی کات نەک تەنها سووڕی ژیانی گەشەپێدان خێراتر دەکات بەڵکو ڕێگە بە گەشەپێدەران دەدات کاتی زیاتر تەرخان بکەن بۆ تاقیکردنەوەیەکی گشتگیر، ئەمەش دەبێتە هۆی گەشەپێدانی بەرنامەی وێبی بەهێزتر و متمانەپێکراوتر.

QALAAI ZANISTSCIENTIFIC JOURNAL
A Scientific Quarterly Refereed Journal Issued by Lebanese French University – Erbil, Kurdistan, Iraq
Vol. (10), No (4), Winter 2024
ISSN 2518-6566 (Online) - ISSN 2518-6558 (Print)

إنشاء حالات اختبار الويب مع تحسين الوقت باستخدام تقسيم الرسم البياني

الملخص:
اختبار تطبيقات الويب هو عملية أساسية في تطوير البرمجيات تصبح أهمية متزايدة في السنوات الأخيرة بسبب تعقيد وطبيعة الويب المتغيرة لتطبيقات الويب، إلى جانب الحاجة لضمان جودتها في سوق تنافسي. تعتبر الطبيعة الديناميكية لتطبيقات الويب من بين الصعوبات الرئيسية في مجال اختبار تطبيقات الويب. ومن الجدير بالذكر أهمية التوازن بين الوقت حتى السوق وجودة البرمجيات. وفي هذا العصر المميز بالتطور التكنولوجي السريع، يظل اختبار تطبيقات الويب تحديًا متجددًا يواجه المطورين. وبالتالي، اقترح هذا البحث نهجًا لتحسين وقت مرحلة الاختبار في دورة حياة تطوير البرمجيات. ضمن هذا الإطار، يتم تمثيل تطبيقات الويب كنموذج رسم بياني، وبعد ذلك يتم تحليله لتسهيل إنشاء حالات الاختبار تلقائيًا. تم تقديم خوارزمية تقسيم الرسم البياني المحدثة وتم عرض فائدتها في تقسيم نماذج الرسم البياني لتطبيقات الويب قيد الاختبار. استنادًا إلى النتائج، يمكن أن يقلل تطبيق النهج المقترح من وقت الاختبار بنحو ٩٠٪. هذا التوفير الكبير في الوقت لا يسرع فقط دورة حياة التطوير بل يمكنه أيضًا تمكين المطورين من تخصيص مزيد من الموارد لاختبار شامل، مما يؤدي إلى تطبيقات ويب أقوى وأكثر موثوقية.