



SACJ - Volume 37 Number 2 - December 2025

South African Computer Journal

ISSN 1015-7999 (print)
ISSN 2313-7835 (online)

December 2025 - Volume 37 Number 2

South African Computer Journal

Volume 37 Number 2, December 2025
ISSN 1015-7999 (print) – ISSN 2313-7835 (online)

A publication of the South African Institute of Computer Scientists and Information Technologists
Open Access: <https://www.sacj.org.za>

Editor-in-Chief
sacj.editor@gmail.com

Prof Katherine Malan
University of South Africa
malankm@unisa.ac.za

Assistant Editor
Prof Ian Sanders
University of the Witwatersrand
dr.id.sanders@gmail.com

Associate Editors

Prof Marijke Coetzee
North West University
marijke.coetzee@nwu.ac.za

Prof Sigrid Ewert
University of the Witwatersrand
sigrid.ewert@wits.ac.za

Prof AURONA Gerber
Stellenbosch University
auronagerber@sun.ac.za

Prof Hugo Lotriet
University of South Africa
lotrihh@unisa.ac.za

Prof Tendani Mawela
University of Pretoria
tendani.mawela@up.ac.za

Prof Deshen Moodley
University of Cape Town
deshen@cs.uct.ac.za

Prof Hanlie Smuts
University of Pretoria
hanlie.smuts@up.ac.za

Prof Terence van Zyl
University of Johannesburg
tvanzyl@uj.ac.za

Editorial Board

Prof Judith M Bishop
Stellenbosch University, South Africa

Prof Anthony Maeder
University of Western Sydney, Australia

Prof Judith van Biljon
University of South Africa

Prof Derrick Kourie
Stellenbosch University, South Africa

Prof Rossouw van Solms
Nelson Mandela University, South Africa

Prof Paula Kotzé
University of Pretoria, South Africa

Prof Irwin Brown
University of Cape Town, South Africa

Prof Sue Conger
University of Dallas, Irving, TX, USA

Production Editor
sacj.productioneditor@gmail.com

Prof Etienne van der Poel
University of South Africa

South African Computer Journal

NOTES FOR CONTRIBUTORS

The *South African Computer Journal* is an accredited specialist academic journal, publishing research articles, technical reports and communications in English in the Computer Science, Computer Systems and Information Systems domains. Its primary target is research of interest in Africa or published by African researchers, but all quality contributions are considered. All research articles submitted for publication are rigorously refereed by independent peer reviewers. The journal publishes original work that is of international stature. The editorial board comprises local and international scholars of high repute. The journal is published online using the *open access* model, making papers accessible in developing countries where funding to subscribe is scarce.

Submissions

Authors should submit papers for publication at the web site at <http://www.sacj.org.za/index.php/sacj/about/submissions>. Please also check there for the latest version of the below guidelines.

Form of Manuscript

Manuscripts for *review* should be prepared according to the following guidelines, which summarize more detailed instructions on the web site.

SACJ has a double-blind reviewing policy. No author's name or affiliation should appear anywhere. Citing of previous work by the author or authors should be anonymised if appropriate. Acknowledgments and thanks should not be included in the draft for review. If you use Microsoft Word please make sure that your name and affiliation are not saved in the document properties.

- The article should start as follows:
 - the title (as brief as possible)
 - an abstract of less than 200 words
 - an appropriate keyword list
 - a list of [ACM Categories \(2012 scheme\)](#).
 - Tables and figures should be numbered, in sequence, titled, and referenced in the text by number.
 - *SACJ* uses American Psychological Association 7th edition style in final, published papers, as described here (<https://apastyle.apa.org/>). Manuscripts for *submission* should use numeric citation, as APA version 7 is difficult to get right, and the numeric citations are easier to process in the production side of the paper. References should be listed at
-

South African Computer Journal

the end of the text in alphabetic order of the (first) author's surname and cited in the text.

If you use an appropriate \LaTeX style, this will work automatically; do not spend a lot of time on reference citation minutiae since the production editor will take care of that sort of detail.

- If a DOI is available for any reference cited, include the DOI in the reference list. The DOI should be a complete URL (ideally clickable), for example: <https://doi.org/10.18489/sacj.v34i1.1115> (DOI display guidelines here: http://www.crossref.org/02publishers/doi_display_guidelines.html)
- If a DOI is not available a URL to the original document or the publication containing the document should be provided.

SACJ is produced using the \LaTeX document preparation system and we recommend using the Overleaf system. A *SACJ* template is available on Overleaf: <https://www.overleaf.com/latex/templates/south-african-computer-journal/smnhsnmsnfdy>. Though we can also accept Microsoft Word submissions, delays in publication are more likely with the latter format. While we encourage submission in a format similar to the published paper, authors should not waste undue time on layout as this will be redone by the production editor.

Authors retain the right to republish their work, subject to any republished version including a reference to the *SACJ* version.

Publication Charges

A charge of R6000 will be levied on papers accepted for publication to cover costs of open access publication. Where the author's institution or research budget is unable to meet this charge, it may be waived upon request of the author and at the discretion of the editor-in-chief.

Proofs

Proofs of accepted papers will be sent to the corresponding author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Extended Conference Papers

Authors of conference papers are welcome to submit extended papers to *SACJ* for consideration on these terms:

- a covering letter accompanying submission should explain what is added to the paper to make it worth publishing as a journal paper
 - the paper includes at least 30% new material
-

South African Computer Journal

- a pointer is provided to the original paper or, if it is not freely available, upload it as supplementary material when submitting the extended paper to *SACJ*
- evidence is provided that republication in this form does not violate copyright of the conference publication

Book Reviews, Letters and Communications

Book reviews are welcome, as are letters to the editor; both should carry the author's full name and affiliation, and should be limited to 500 words. Communications and Viewpoints up to two pages in length (or longer, by negotiation with the editor-in-chief) may also reflect minor research contributions, works-in-progress or ideas to stimulate debate.

This category of submission is accepted at the discretion of the editor-in-chief, not refereed and does not qualify as a research publication for South African government subsidy purposes. The major criteria for acceptance are that the item is coherently written and does not require significant editing, that it is timely and it is likely to be of interest to readers.

South African Computer Journal


Volume 37 Number 2, December 2025
ISSN 1015-7999 (print) – ISSN 2313-7835 (online)

A publication of the South African Institute of Computer Scientists and Information Technologists
Open Access: <https://www.sacj.org.za>

CONTENTS

Editorial: The international issue	vii
<i>Katherine Malan</i>	
 Obituary – André van der Poll	1
<i>Jan H. Kroeze</i>	
 Obituary – Derek Smith	3
<i>Irwin Brown, Lisa Seymour, Michael Hart, and Michael Eccles</i>	
 General Research	
A 45-year review of the South African Computer Journal (1979–2023)	5
<i>Filistea Naudé and Jan H. Kroeze</i>	
 Investigating the support of proprioception and visual guidance for menu selection in virtual reality	37
<i>Kwan Sui Dave Ka, Isak de Villiers Bosman, and Theo J.D. Bothma</i>	
 Improving greybox fuzzing with dictionary-based mutations: A systematic literature review	74
<i>Enock L. Dube, Boluwaji A. Akinnuwesi, Stephen G. Fashoto, Petros M. Mashwama, and Vusi W. Tsabedze</i>	
 WeaveChain: A decentralized storage framework using Arweave to address scalability, Security, and Decentralization Challenges	104
<i>Saha Reno</i>	
 Reinforcement Learning algorithms for adaptive load-balancing for Web applications	121
<i>Rana Zuhair Al-Shaikh, Muna M. Jawad Al-Nayar and Ahmed M. Hasan</i>	

Editorial: The international issue

Katherine M. Malan  – sacj.editor@gmail.com

Department of Decision Sciences, University of South Africa

In this issue

We start SACJ volume 37, issue 2, with tributes to the lives of two distinguished academics in the SACJ community, Prof André van der Poll and Prof Derek Smith. With obituaries from colleagues, we celebrate their lives and significant contributions to both the Computer Science and Information Systems communities of South Africa. They will be sorely missed.

The research in this issue is distinguished from previous issues of SACJ in the high number of research articles from international scholars. Of the five research articles, only two are from South African authors, with the other three articles from Eswatini, Bangladesh and Iraq.

The first research article by Naudé and Kroeze is a bibliographic analysis of SACJ spanning 45 years, providing interesting insights on publication numbers, citation count and co-authorship trends over 45 years of SACJ's history. To quote from the conclusion: "*SACJ has played a pivotal role in developing and disseminating computing knowledge in South Africa.*" With this 'international issue', we see the role of SACJ expanding beyond the borders of South Africa. We welcome our new international authors.

The research papers included in this issue are:

- [A 45-year review of the South African Computer Journal \(1979–2023\)](#) by Naudé and Kroeze.
- [Investigating the support of proprioception and visual guidance for menu selection in virtual reality](#) by Ka, Bosman, and Bothma.
- [Improving greybox fuzzing with dictionary-based mutations: A systematic literature review](#) by Dube, Akinnuwesi, Fashoto, Mashwama, and Tsabedze.
- [WeaveChain: A decentralized storage framework using Arweave to address scalability, security, and decentralization challenges](#) by Reno.
- [Reinforcement Learning algorithms for adaptive load-balancing for Web applications](#) by Al-Shaikh, Al-Nayar, and Hasan.

Malan, K.M. (2025). Editorial: The international issue [Editorial]. *South African Computer Journal* 37(2), vii–viii. <https://doi.org/10.18489/sacj.v37i2.25176>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](#) 
SACJ is a publication of [SAICSIT](#). ISSN 1015-7999 (print) ISSN 2313-7835 (online)

Submission statistics

For information, we present the following updated statistics around submissions, acceptance rates and the processing time of articles.

Year	2022	2023	2024	2025
Submissions	111	95	96	116
Desk rejection	54	62	56	
Desk rejection rate	49%	65%	58%	
Final acceptance	17	12	11	
Acceptance rate	15%	13%	11%	

The figures above translate into an overall rejection rate of approximately 87%. This high rejection rate can be attributed to the large number of submissions from all over the world due to the indexing of SACJ on Scopus. To quantify the reach of SACJ, between 1 January and 20 December 2025 we received 116 submissions from 27 different countries (36 from South Africa, 25 from other African countries, and 55 from outside Africa).

Similar to SACJ, South African Journal of Science reported a rejection rate of 82% in 2024. In terms of submission processing time, the median time to a first decision (desk reject / send to review) since 2023 was 2 days and the median time to final decision for submissions sent to review was 5 months.

Thank you to all authors for submitting your manuscripts to SACJ and for entrusting us with your research. Thank you also to the editorial team for all your work behind the scenes and to the many reviewers who support our journal.

Obituary – André van der Poll

Jan H. Kroeze^a – jan.kroeze@gmail.com, kroezjh@unisa.ac.za

Department of Information Systems, School of Computing, University of South Africa, Roodepoort, Johannesburg, South Africa

It is with great sadness that we need to inform you that Prof André van der Poll, research professor in the Graduate School of Business Leadership and formerly professor in the School of Computing at the University of South Africa, passed away in the early morning of 17 October 2025.

John Andrew (André) van der Poll was born on 28 March 1960 in Nababeep, Northern Cape, South Africa. He matriculated at Brackenfell High School in the Western Cape province, South Africa in 1977 with an average of 80%. John did a B.Sc. degree at the University of Stellenbosch from 1978 to 1980 and majored in Computer Science, Mathematics and Applied Mathematics. He enrolled for an Honours B.Sc. degree in 1981 at the same institution, which he completed the following year.

After his compulsory military service of two years, he took up a position in 1984 at the Data Processing Centre of the South African Post Office (now Telkom). He was employed first as a systems developer and later as a training officer. During his time at Telkom, he enrolled for an M.Sc. degree at Unisa (University of South Africa) and completed the degree in 1987. The title of his dissertation was “Towards the Denotational Semantics of QuadLisp’86”.

John (we used to call him André, by his middle name) was appointed as a senior lecturer in the then Department of Computer Science and Information Systems at Unisa in January 1988. He completed a PhD degree in Computer Science at Unisa in June 2000, and the title of his



Kroeze, J.H. (2025). Obituary – André van der Poll [Obituary]. *South African Computer Journal* 37(2), 1–2. <https://doi.org/10.18489/sacj.v37i2.25094>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](#) 
SACJ is a publication of [SAICSIT](#). ISSN 1015-7999 (print) ISSN 2313-7835 (online)

thesis was “*Automated Support for Set-Theoretic Specifications*.” During the fall term of 2000, he was a visiting professor at the University of McMaster in Hamilton, Canada. He was promoted to the position of associate professor in January 2002 and to full professor in January 2006. During his time at Unisa, he taught courses in Discrete Mathematics for Computer Science, Compiler Construction, Formal Logic, Formal Program Verification, Automated Reasoning, Formal Specification Techniques, Computer Networks, Database Systems, Software Project Management and Operating Systems.

In July 2013, he accepted a position as Professor in ICT Management at Unisa’s Graduate School of Business Leadership (SBL) where he offered modules for company executives in Business Information and Communication Technologies (ICTs). His research interests extended to include the advancement of the use of formal methods for business software development among upper management in the industry. He was the Academy for Global Business Advancement (AGBA) Vice President for South Africa, Northern region. He delivered numerous MBA/MBL students with a variety of dissertation topics in Business ICTs.

John was the lead author and co-author of more than 120 journal articles and peer-reviewed conference papers. He was the guest editor of a special edition of the Journal for Global Business Advancement (JGBA), from the Inderscience publisher. Most of his journal articles are indexed in the Web of Science (WoS), Scopus and the Directory of Open Access Journals (DOAJ). He delivered more than 24 Master of Science (MSc), Master of Business Administration (MBA), Master of Business Leadership (MBL), Doctor of Philosophy (PhD), Doctor of Business Leadership (DBL), and Doctor of Commerce (DCom) students.

His research interests were in the use of Computing formalisms in software development for, amongst other, Business ICTs. He believed that it is essential that computer systems function as expected, especially safety-critical systems that have human lives at stake. Using mathematical techniques to produce reliable software can limit problems and create highly dependable systems. Formal methods can be used to predict the behaviour and consequences of a system and assist in embedding safe, ethical principles into the algorithms of software, thus preventing malicious behaviour.

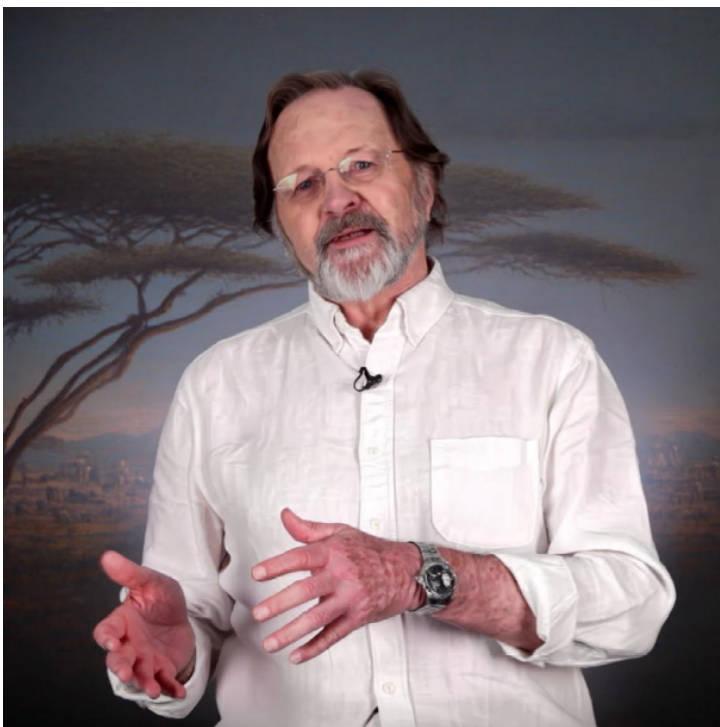
Prof van der Poll was an NRF rated researcher, category C2, and a Research Professor at Unisa’s SBL. He held a three-year NRF Competitive Program for Rated Researchers (CPRR) grant for the period 2023 to 2025.

Many of us in the South African Computing community have known him and collaborated with him over many years. He was a kind spirit who always saw the potential in other colleagues and students. He had an exceptional gift to develop postgraduate students’ research skills and nurture their critical thinking abilities. As a supervisor, he always persevered and successfully delivered many master’s and doctoral students. Above all else, he was a wonderful colleague and friend with a subtle and charming sense of humour. He was a petrolhead who loved his BMW750i 2008 and always told his friends stories about his car. We will miss him dearly. We would like to extend our condolences to his wife, Breggie, their children, Hendrik and Saretha, and family.

Obituary – Derek Smith


Irwin Brown – Irwin.brown@uct.ac.za,
Lisa Seymour – lisa.seymour@uct.ac.za,
Michael Hart – mike.hart@uct.ac.za,
Michael Eccles – meccles@iafrica.com

Derek Smith a distinguished academic, visionary leader, and cherished colleague whose contributions shaped the field of Information Systems in South Africa died on the 3rd December 2025. He was born in Durham, England on the 7th December 1945, and completed his schooling at Kings School, Pontefract, West Yorkshire. Derek began his career after earning a BSc in Chemistry from Bradford University, Yorkshire, joining Rolls Royce in 1968 as a Systems Programmer. In 1970, he moved to Zambia, serving as IT Operations Manager for Roan Consolidated Mines for five years. In 1976, he relocated to Cape Town, where he worked as Systems Development Manager at Shell.



In 1983, Derek enrolled for an MCom in Information Systems (IS) at the University of Cape Town (UCT). During his studies, he transitioned from industry to academia, joining UCT as a Senior Lecturer. At that time, the IS Section was part of the Accounting Department. Derek's leadership was instrumental in transforming the section into a standalone Department of Information Systems as from 1994, where he served as its first Head of Department. Under his guidance, the department grew from five staff members into one of South Africa's leading IS departments, expanding its offerings from an undergraduate focus to postgraduate diplomas, honours, master's, and PhD programmes, and establishing a strong research culture. He retired as emeritus professor at the end of 2010.

Brown, I, Seymour, L, Hart, M, and Eccles, M. (2025). Obituary – Derek Smith [Obituary]. *South African Computer Journal* 37(2), 3–4. <https://doi.org/10.18489/sacj.v37i2.25171>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](https://creativecommons.org/licenses/by-nc/4.0/) 
SACJ is a publication of [SAICSIT](https://www.saicsit.org/). ISSN 1015-7999 (print) ISSN 2313-7835 (online)

Derek's academic passions lay in project management and the IS Honours programme, which has become a cornerstone qualification for aspiring IS professionals. His teaching and mentorship influenced countless IT leaders and entrepreneurs, many of whom credit Derek for inspiring them. He fostered strong ties between academia and industry, a legacy that continues today. In addition, he promoted community outreach being instrumental in establishing the Honours Outreach and Community Involvement Programme (HOCIP) in 2004.

Beyond UCT, Derek co-founded the Faculty Training Institute in 1989, which has grown into one of South Africa's largest training organisations in Business Analysis and Project Management. He also played a pivotal role in developing UCT project management short courses in partnership with GetSmarter (now a brand of 2U) and developed the first online 12-month Advanced Diploma in Business Project Management which was completed by over 1000 students. He contributed to multiple communities, lecturing at TSiBA and on the Tutu Leadership Programme and was Ministerial advisor to the South African Department of Education in 2006.

As a researcher, Derek published extensively on project management and the human aspects of IS, contributing to leading local and international journals and conferences. He was an active member of the South African Institute for Computer Scientists and Information Technologists (SAICSIT) community and a regular supporter of the Southern African Computer Lecturers' Association (SACLA). In recognition of his impact, he received the Pioneer Award from SAICSIT in 2012.

Outside academia, Derek had a lifelong love of cricket and rugby. He was a strong rugby forward in teams in England, Zambia, and then at Villagers in Cape Town. He was a regular at Newlands and even combined his expertise in team dynamics with his passion for rugby when invited to deliver a motivational talk to the Western Province rugby team. Colleagues and former students remember him for his warmth, generosity, and unwavering support. He was a tall man yet gentle. He loved the African bush and had a special affinity with the gentle giants of the bush – the African elephants and in his later years loved watching Safari Live. He loved a good glass of red wine, was especially fond of one at a bargain and was well loved in his community, fondly referred to as his village.

Derek's vision and dedication helped shape the Department of Information Systems into what it is today. We are profoundly grateful for his contributions. He is survived by his beloved wife Liz, his children Rachel and Simon, and his grandchildren.

A 45-year review of the South African Computer Journal (1979–2023)

Filistéa Naudé , Jan H. Kroeze 

University of South Africa, Christiaan de Wet Road, Florida Park, Roodepoort, South Africa

ABSTRACT

This single-journal study provides a comprehensive overview of the history and development of the *South African Computer Journal* (SACJ; formerly *Quaestiones Informaticae*), combined with a bibliometric and altmetric analysis spanning 45 years (1979–2023). Citation data were sourced from Google Scholar and Scopus, while Mendeley readership counts were used as an alternative metric indicator. In total, 768 articles were analysed to assess journal productivity, authorship patterns and collaboration trends. On average, 17 articles were published per year, with an average of 8.3 Google Scholar citations per article. Citation data revealed that 60% of articles had Google Scholar citations, while 40% remained uncited. The journal exhibited a notable shift from single to multi-authorship, with co-authored articles increasing from 22% (1979–1989) to 70% (1990–2023). The results indicate a strong association between collaboration and scholarly impact, as articles with two or three authors consistently achieved higher citation counts and Mendeley readership. The most productive authors and most cited papers were identified. The SACJ journal metrics were also compared with selected international journals in the Computing discipline.

Keywords Altmetrics, Authorship, Bibliometrics, Citation analysis, Collaboration, Google Scholar, Mendeley readership, *Quaestiones Informaticae*, Scopus, *South African Computer Journal*

Categories • Applied Computing ~ Digital libraries and archives • General and reference ~ Surveys and overviews

Email

Filistéa Naudé – fnaude@unisa.ac.za
Jan H. Kroeze – jan.kroeze@gmail.com (CORRESPONDING)

Article history


Received: 11 April 2025
Accepted: 5 September 2025
Online: 22 December 2025

1 INTRODUCTION

The scientific community primarily relies on expert judgement or peer review as the key qualitative method for research assessment. Bibliometric citation-based indicators serve as quantitative measures used to evaluate scholarly productivity and impact. These metrics are typically applied alongside, or in support of, peer review. Citation-based indicators not only reflect the current impact of research, but are also considered useful predictors of future influence (Abramo et al., 2019).

Scholarly impact is commonly measured by the number of citations an article or journal receives. Citation analysis is a bibliometric method used to quantitatively assess research per-

Naudé, F., Kroeze, J.H. (2025). A 45-year review of the South African Computer Journal (1979–2023) . *South African Computer Journal* 37(2), 5–36. <https://doi.org/10.18489/sacj.v37i2.21924>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](https://creativecommons.org/licenses/by-nc/4.0/) 
SACJ is a publication of *SAICSIT*. ISSN 1015-7999 (print) ISSN 2313-7835 (online)

formance, including aspects such as productivity, citation impact and research quality (Garousi & Fernandes, 2017). Donthu et al. (2021) define the bibliometric methodology as “*the application of quantitative techniques (i.e., bibliometric analysis, citation analysis) on bibliometric data (e.g., units of publication and citation)*”.

Due to the limitations of traditional bibliometric indicators, altmetrics (or alternative metrics) were developed to provide additional quantitative measures for research assessment. Altmetrics capture broader aspects of research impact, including non-academic and societal influence, and are intended to complement traditional citation-based indicators (Thelwall, 2020). Altmetrics “*measure the digital attention that an article receives by using data from different online resources*” (García-Villar, 2021). One of the key sources of altmetric data is online reference managers such as Mendeley (Araujo et al., 2021). Mendeley readership has been identified as particularly relevant for research evaluation, as reader counts have been found to correlate moderately to strongly with citation counts across most academic disciplines (Thelwall, 2020).

This study presents a 45-year (1979–2023) overview of the *South African Computer Journal* (SACJ). After an overview of the journal’s history, the bibliometric analysis identifies some of the leading trends in terms of publication and citation behaviour, authorship, collaboration, the most productive authors and highly cited papers.

This article is organised as follows: Section 2 states the aim and objectives of this undertaking. Section 3 provides an overview of the journal’s history and development, while Section 4 reviews the relevant literature and previous studies. Section 5 outlines the methods used in the article. Section 6 presents the results, while Section 7 summarises and discusses the main findings. Section 8 presents the article’s conclusions.

2 AIM AND OBJECTIVES

According to Kotzé and van der Merwe (2009), SACJ plays a pivotal role in South Africa, as few local journals accredited by the Department of Higher Education and Training (DHET) are available for Computer Science scholars to publish their research. Offering publication opportunities in reputable journals is essential, given that journal articles tend to attract more citations than conference papers (Parry, 2019). As a core publication outlet for the South African research and academic community in Computer Science and Information Systems, SACJ plays an important role in disseminating local scholarship. A bibliometric analysis of the journal can offer valuable insights into the nature of, and trends in, Computing¹ research in South Africa, while also shedding light on the journal’s quality, credibility and influence. Such an analysis may help to further strengthen the standing of SACJ in the scholarly community and guide its future development.

¹ *Computing* is used as a catch-all umbrella term for various related disciplines – see the clarification of the terminology at the end of Section 2.

The following objectives informed our endeavour to realise this aim:

- Provide an account of the history of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) and *SACJ* (see [Section 3](#))
- Summarise and synthesise the relevant literature (see [Section 4](#))
- Design a suitable methodology to analyse *SACJ*'s bibliometrics (see [Section 5](#))
- Document the journal's productivity (see [Section 6.1](#))
- Compare the Google Scholar citations, Scopus citations and Mendeley readership scores (see [Sections 6.2](#) and [6.3](#))
- Assess authorship patterns and collaboration trends (see [Section 6.4](#))
- Highlight the most influential articles based on citation counts and readership metrics (see [Section 6.5](#))
- Identify the most productive authors based on article productivity (see [Section 6.6](#))
- Determine to what extent articles in a regional South African Computing journal from the developing world are cited (see [Sections 6.2](#) and [6.7](#))
- Benchmark and compare *SACJ* to selected international journals, using Scopus journal metrics (see [Section 6.7](#))

Importantly, “Computer Science” is often used as an umbrella term encompassing the broader field of Computer Sciences. For example, the DBLP Computer Science Bibliography (DataBase systems and Logic Programming, a metadata repository) (DBLP, [2025](#)) includes Information Systems journals in its database. There is also significant thematic overlap between master's and doctoral research in Computer Science and Information Systems (Sanders & Alexander, [2015](#)). Thus, in the remainder of this article, Computer Science is used to refer broadly to the wider field of Computing, with the terms being used interchangeably, in line with the ACM (Association for Computing Machinery) Computing Classification System (CCS), which uses “Computing” in this broader sense (ACM-CCS, [2025](#)).

3 BRIEF OVERVIEW OF THE HISTORY OF *SACJ*

This section provides more information on the history of *SACJ* and the academic society that owns the journal. First, the focus falls on the academic society (SAICSIT) and its conferences, before shifting to *SACJ*. Thereafter, the discussion moves to the relationship of SACLA and its conferences with SAICSIT and *SACJ*.

3.1 THE HISTORY OF SAICSIT

In the late 1970s, SAICSIT grew out of a professional information technology (IT) guild, the Computer Society of South Africa (CSSA). The CSSA's name later changed to the Institute of Information Technology Professionals South Africa (IITPSA), the professional body for information and communication technology (ICT) practitioners (IITPSA, 2025).

In those early days of industrial computing, the CSSA had an academic wing, namely the Research Symposium Organising Committee. Dr Phil Roets, who had been involved in the publication of the symposium proceedings right from the start in 1979, conducted postdoctoral Computer Science studies from 1970–1973 at Cornell University in New York.

Starting soon after his return from the USA, several symposia were organized by a number of University lecturers, with involvement by NRIMS [the National Research Institute for Mathematical Sciences] and the CSIR [the Council for Scientific and Industrial Research in South Africa]. After the second such symposium [28–29 October 1981], SAICS [the South African Institute of Computer Scientists] was founded as a separate entity from the CSSA by a steering committee that included Phil as one of its members.

[Taken from Phil Roets's biography – see Supplement A in Naudé and Kroeze (2025)]

According to Kotzé (see Supplement B in Naudé and Kroeze (2025)), SAICS was formed in 1982. In the same year, the new academic society effectively took over the activities of the CSSA's symposium committee. In 2005, in the preface of the 20th conference proceedings, Prof Judith Bishop provided a brief recollection of the founding of the academic society:

In 1982, twelve good men and true (including one woman) got together over a (few) bottles of wine and decided to form a scientific body dedicated to promoting the interests of computing research in this country. The people were: Stef Postma (RAU) [Rand Afrikaans University], Judith Bishop (Wits) [University of the Witwatersrand], Pierre Visser, Gideon De Kock (UPE) [University of Port Elizabeth], Niek du Plooy (CSIR), Doug Laing (IBM), Ken MacGregor (UCT) [University of Cape Town], Phil Roets (NRIMS), Trevor Turton (IBM), Gerrit Wiechers (UNISA) [University of South Africa], Trevor Winer, Roelf van den Heever (UP) [University of Pretoria] and Derrick Kourie (UP). The enduring legacy of this group of legends in computing has been the SA [South African] Computer Journal, and the Annual SAICSIT conference.

[See Supplement C in Naudé and Kroeze (2025)]

In the first 16 years of the society (1979–1994), as the research committee of the CSSA and later SAICS, at least seven computer research symposia were offered – see Supplement D in Naudé and Kroeze (2025) for a list showing the numbers, years and names of all the conferences (1979–2024).

In 1995, after much debate, the name of the SAICS was changed to SAICSIT as an independent academic society catering for scholars in Computer Science and Information Systems, as well as other related Computing disciplines, and the organisation's constitution was revised as well.

From 1995 onwards, the SAICSIT conference has been held annually, without interruption, following its formative years as the CSSA/SAICS Computer Symposium. In 2025, with the 31st SAICSIT Conference, the society celebrated 30 years of the more inclusive version of the society (1995–2025) and running its annual conferences without interruption, as well as its 46th birthday and 38th conference, if the CSSA/SAICS computer symposia are counted, and assuming that no conferences were held in 1993 and 1994 (see below) (see Supplement D in Naudé and Kroeze (2025)).

The proceedings of the first three CSSA research symposia (1979, 1981, 1983) were published in a new journal, *Quaestiones Informaticae (QI)*, with SAICS becoming involved in 1983. The next four proceedings (1987, 1989, 1991 and 1992) were jointly published by the CSSA and SAICS – with the International Federation for Information Processing (IFIP) (IFIP, 2025) also involved in 1987 and 1989 – as standalone books, except for the 1992 issue, which appeared in *SACJ*, number 7.

It is unclear whether SAICS symposia were held in 1993 and 1994, as no records or proceedings from those years could be located. Although Prof Bishop mentioned in the preface of a later SAICSIT conference proceedings (2005) that this was the 20th conference (including the CSSA and SAICS conferences), no concrete evidence could be found that conferences were indeed held in 1993 and 1994, despite an extensive search for published papers or proceedings (see Supplements C and D in Naudé and Kroeze (2025)). It is, however, possible that conferences were held and that the papers, or revised versions thereof, were published in *SACJ* (or other journals), as was done with the first three symposia.

Since 1995, the annual SAICSIT conference proceedings have been published without interruption. From 1995–1998, the proceedings were printed as standalone books. In 1999 and 2000, special editions of *SACJ* (24 and 26) were used for this purpose. In 2001, SAICSIT went back to publishing standalone volumes – a tradition upheld until 2020. From 2002–2020, the annual SAICSIT proceedings all appeared in the ACM Digital Library (ACM, 2025) in electronic format, while printed copies were still made available for some time. In 2021, the tradition of publishing with ACM came to an end, and the proceedings appeared as an e-book (Singh et al., 2021); in 2022, it appeared in the EPiC (EasyChair Proceedings in Computing) series (Gerber, 2022). Selected revised papers of the 2023 conference appeared in Springer’s *Communications in Computer and Information Science (CCIS)* (Gerber & Coetzee, 2023a), while other peer-reviewed, accepted papers appeared in their own series as “Online Proceedings” (ISSN² = 2959-8877) (Gerber & Coetzee, 2023b), available on the SAICSIT Conference website. The 2024 and 2025 conferences followed the same publication model (Gerber, 2024a, 2024b, 2025a, 2025b).

The paragraphs above gave a brief overview of the history of SAICSIT and its annual conference proceedings. For more details on the SAICSIT conferences, some proceedings of which appeared in *QI* or *SACJ*, see Supplements D, E and F in Naudé and Kroeze (2025). A list of the founding members and presidents of SAICSIT is available in Supplement G in Naudé and Kroeze (2025).

² ISSN = International Standard Serial Number.

3.2 THE HISTORY OF SACJ

In this section, we provide a brief overview of the emergence of *SACJ* from *QI*. The origin of the journal can be traced back to the series of research seminars organised by the Research Symposium Organising Committee of the CSSA, discussed above.

Kourie (2010) documents the historical development of the journal. *QI* was established in 1979 (ISSN=0254-2757). The proceedings of the first three symposia were published in *QI* (the CSSA's journal, ISSN=0254-2757), but the journal also accepted other submissions. Between 1979 and 1989, a total of 19 issues in six volumes appeared.

In 1990, *QI* was renamed as *SACJ* (online ISSN = 2313-7835; print = 1015-7999) to be more descriptive and target a wider audience (see Supplement H in Naudé and Kroeze (2025) and Kourie (1989)). Notably, the phrase *Computer Science and Information Systems* appears as a subtitle on the first edition of *SACJ*, a peer-reviewed scholarly journal owned and published semi-annually by SAICS, and later by SAICSIT (2025), with additional special editions.

The relationship of *SACJ* to the SAICSIT conference has varied over the years. The journal's predecessor, *QI*, belonged to the CSSA, as did *SACJ* originally, but SAICS and SAICSIT were also involved in the editorial process. From 1995, SAICSIT assumed ownership of, and responsibility for, *SACJ*. Both journals published some proceedings, along with selected or revised conference papers and other submissions. For example, a separate proceedings book for the 1989 symposium exists, but some of the papers were also published in the first two issues of *SACJ* (1990).

The journal publishes original research articles and shorter technical research notes in English in the subject fields of Computer Science, Computer Systems, Information Systems and related fields. The geographic focus is on South Africa, the rest of Africa and less-developed countries, but other international submissions are also considered. The journal also publishes viewpoint articles, letters to the editor, book reviews and announcements. Its editorial board comprises local and international scholars.

For a summary of the availability of the digitised and electronic proceedings of all the SAICSIT conferences and their predecessors – South African Computer Symposium (SACS), South African Computer Research Symposium (SARCS), and SAICS – see Supplement D in Naudé and Kroeze (2025).

3.3 THE HISTORY OF SACLA

Since the chair of the Southern African Computer Lecturers Association (SACLA) (SACLA, 2025) serves ex officio on the SAICSIT Council, there has been a close relationship between the two societies for many years, and the two annual conferences have sometimes been offered back to back. The 2008 SACLA proceedings appeared in a special edition of *SACJ* (Kourie, 2010). The 50-year history and development of SACLA and the Computer Science departments in South Africa are described by Calitz (2022).

3.4 AVAILABILITY AND VISIBILITY OF THE PROCEEDINGS AND JOURNAL ISSUES

All the historical print-only editions of *QI*, *SACJ* and SAICSIT proceedings were digitised in 2018 (see Supplement I in Naudé and Kroeze (2025)). When the SAICSIT Council embarked on the endeavour to digitise previously print-only issues of its journal and conference proceedings (see Supplement I in Naudé and Kroeze (2025)), the goal was to enhance the visibility and availability of the 1979–2001 *QI/SACJ* issues and SAICSIT proceedings. It was hoped that, over time, this would lead to increased citation rates for these older papers and articles. The electronic copies of all the papers and articles were made available in the SAICSIT Digital Archive, hosted on the Unisa Institutional Repository (UIR) (SAICSIT, 1979–2021) (see Supplement F in Naudé and Kroeze (2025))³. Kotzé and van der Merwe (2009) note that, up to 2009, *SACJ* had a low citation rate, possibly because it was not an open-access journal.

For a summary of the availability of the digitised and electronic proceedings of all the SAICSIT conferences and their predecessors – SACS, SARCS and SAICS – see Supplement F in Naudé and Kroeze (2025). For an overview of all the papers published in *SACJ*, see *SACJ* (2009–2023, 2021–2025) and SAICSIT (1979–2021). Unfortunately, no complete collection of the printed conference proceedings is available, but partial collections do exist (see Supplement F in Naudé and Kroeze (2025)) (note: there may be more scattered issues in other South African libraries).

Hard copies of the early, print-only issues of *SACJ* were donated to the university libraries of the North-West University (NWU) and the University of Pretoria (UP) (which now have complete collections of *QI/SACJ*), as well as the University of Cape Town (UCT) and the University of the Free State (UFS) (which have partial collections).

SACJ is accredited by DHET (2015) and is included in four of the 2025–2026 DHET-accredited publication lists (Sabinet, 2025a), namely DHET’s sublist of Scopus (Elsevier, 2025), the Directory of Open Access Journals (DOAJ, 2025), the Scientific Electronic Library Online South Africa (SciELO SA) (SciELO, 2025), and DHET’s own list of local accredited journals (DHET, 2025).

QI (1979–1989) and *SACJ* (1990–2009) were indexed in the IET Inspec database, a subscription-based resource compiled by the Institution of Engineering and Technology (IET, 2025).

Since 2000, *SACJ* has been available in electronic format on Sabinet⁴ (Sabinet, 2025b) (cf. Kotzé and van der Merwe (2009)) in the Science, Technology and Agriculture Collection of the African Journals database. Until 2009, the journal was only accessible through these specialised paywalled databases.

Its discoverability significantly improved, however, from 2009, when *SACJ* became an open-access journal, published on the Open Journal Systems platform (*SACJ*, 2009–2023). In

³ The 1989 proceedings had been missing, but after the peer review process, Prof Martin Olivier discovered a copy in his library and donated it to the SAICSIT Council, to be scanned and added to the printed and digital collections (to be done).

⁴ South African Bibliographic and Information Network.

December 2022, *SACJ* joined Khulisa Journals under the auspices of the Academy of Science of South Africa (ASSAf, 2025). The journal's website was migrated to a new platform (*SACJ*, 2021–2025). The issues that appeared from 2009 are also available on the journal's previous website (*SACJ*, 2009–2023), and will remain accessible for some time (Malan, 2023). Several other free web-based services also index *SACJ*, notably SciELO (2025), DBLP (2025), Dimensions (2025) and DOAJ (2025).

The next section moves from the history of *QI* and *SACJ* to a literature review focusing on earlier studies of the journal and other related research.

4 LITERATURE REVIEW

A review of the South African literature reveals that only three previous studies (Kotzé & van der Merwe, 2009; Kourie, 2010; Machanick, 2024) specifically focused on *QI* and *SACJ*. A brief overview of these is provided below.

Kotzé and van der Merwe (2009), who analysed the research topics of articles published in *SACJ* from 1990–2008, examined 344 research articles. In addition, they identified the most productive South African higher education and research institutions and examined publication trends across subject categories and subcategories. However, their study did not include a citation analysis or other bibliometric data, nor did it cover *QI*.

Kourie (2010) provides a historical profile of *QI* and *SACJ*, detailing the journal's founding, editorial leadership and the roles of the CSSA and SAICSIT. The article highlights the ten most-cited articles from 1989–2010, based on Google Scholar citation scores. The study emphasises the need for marketing to enhance *SACJ*'s readership, visibility and citation impact. Also noted (Kourie, 2010), *SACJ* issues 7, 9, 13, 19, 22, 24 and 26 were special editions that published conference or workshop proceedings, while issues 17, 40 and 42 contained conference papers that were significantly updated and peer reviewed once more. For further details on *SACJ*'s editorial history, see Supplement J in Naudé and Kroeze (2025).

Machanick (2024) provides an overview of his experiences as editor-in-chief of *SACJ*, focusing on how he developed it into a quality journal with a developmental and mentorship agenda for emerging academics in developing countries. Machanick's experience demonstrated that it is possible to establish a quality, regionally focused journal that bridges the gap between predatory and prestigious publications, highlighting the critical need for a mentoring model for editors to maintain such standards. Notably, in the ecosystems of scholarly journals *SACJ* fulfils a unique niche role, which may hamper its competitiveness:

Competing in the middle ground of a quality journal with a developmental role is not easy. Authors who aspire to publish in top-ranked journals will go there and authors who require quick publication to check a box want guaranteed turnaround time; that can only be achieved with either a high reject rate or giving up on quality. [Machanick (2024)]

A broader analysis of the South African Computing research literature, beyond *QI* and *SACJ*, revealed only a handful of studies (ASSAf, 2019; Brown & Tanner, 2008; Mouton et al., 2022; Mouton et al., 2019; Naudé & Kroeze, 2024; Parry, 2019; Turpin, 2018; van Biljon & Naude, 2018) on the South African research landscape.

Brown and Tanner (2008) investigated the international visibility of South African Information Systems research by analysing publications from South African-affiliated authors in 50 Information Systems journals between 2003 and 2007. They found that only 19 of these journals contained articles with South African authorship, totalling just 39 articles. The study highlights the low international visibility of South African Information Systems research and suggests that the high-ranking Information Systems journals are not well suited to South Africa's development-oriented research focus.

van Biljon and Naude (2018) analysed the research collaboration patterns among South African ICT4D (Information and Communication Technology for Development) researchers from 2003 to 2016, based on publications in *Information Technologies and International Development (ITID)*, *the Electronic Journal of Information Systems in Developing Countries (EJISDC)* and *Information Technology for Development (ITD)*. The findings revealed that most articles were co-authored (78%): collaboration patterns showed that intra-institutional collaboration (51%) was the most prevalent, followed by international (35%) and inter-institutional collaboration (14%).

Turpin (2018) studied 58 articles published between 2006 and 2015 by South African ICT4D researchers in *ITID*, *EJISDC* and *ITD*, examining publication patterns across institutions, geographic regions and research domains. The findings showed a preference for collaboration among South African authors, with 53 of the 58 papers being co-authored. Of the 53 papers, 18 had international collaborators and 28 had co-authors from the same institution.

Parry (2019) conducted a comprehensive scientometric investigation of Computing research in South Africa, using the Elsevier Scopus database. That study investigated the journal articles, books, book chapters and conference proceedings of researchers affiliated with South African universities from 2008 to 2017 and assessed annual production, differences in institutional outputs, topical trends, collaboration and citation impact. The findings showed a 235% increase in Computing research over the decade under study. Conference papers were the most popular publication format (61.40%), followed by journal articles (36.10%), book chapters (2.33%) and books (0.23%). The study noted that *SACJ* did not appear among prominent publications due to its late inclusion in Scopus (post-25 March 2016), limiting the analysis to *SACJ* data from 2016 and 2017.

The ASSAf Committee on Scholarly Publishing conducted a grouped peer review of South African scholarly journals in the Communication and Information Sciences (ASSAf, 2019), and included *SACJ* in this review. The report highlights *SACJ*'s unique role in the South African Computing research landscape, recognising its high-quality articles across diverse topics (ASSAf, 2019).

Naudé and Kroeze (2024) investigated the research participation of South African-based authors in Springer's *Lecture Notes in Computer Science (LNCS)*, a conference proceedings book

and e-book series, for the period 1973–2022. The study compared these authors' publication output and citation impact to the global Computer Science and *LNCS* output trends. The findings highlight a strong collaborative publication culture among South African *LNCS* authors, with 91% of articles co-authored and 9% single-authored. Of the articles, 73% had received citations, while 27% remained uncited. When the citation metrics of *LNCS* were compared to *SACJ*, *LNCS* consistently outperformed *SACJ*. In 2022, *LNCS* achieved a CiteScore⁵ of 2.2, compared to *SACJ*'s 0.9. Similarly, *LNCS* recorded a SCImago Journal Rank score⁶ of 0.320, exceeding *SACJ*'s score of 0.170. Additionally, *LNCS* attained a Source Normalised Impact per Paper (SNIP) score⁷ of 0.542, while *SACJ* obtained a score of 0.314.

Mouton et al. (2019) undertook a large-scale, multi-disciplinary empirical assessment of South Africa's research performance (including research output, collaboration and citation impact). The bibliometric analyses were based on two databases: the Clarivate Web of Science and the South African Knowledgebase (SAK), a database of university research publications compiled by the Centre for Research on Evaluation, Science and Technology of Stellenbosch University. Section 4.6 of the report covers an analysis of Computer Science from 2000 to 2016. During this period, South African researchers produced a total of 1716 articles in the field of Computer Science. The leading institutions by publication volume were UP (416 articles), UCT (268), US (256) and Wits (227).

A scientometric assessment of Computer Science in South Africa (in its wider sense as Computing) was conducted (Mouton et al., 2022) for the period 2005–2020. The findings identified *SACJ* as the second most popular publication outlet for local Computer Science researchers, following *LNCS*. This study analysed 3441 articles in 472 publications retrieved from the SAK database of DHET subsidy-earning publications. The adapted data (Table 1) underscore the journal's prominence as a key publishing platform within the discipline (see Supplements K and L in Naudé and Kroeze (2025)).

The literature review clearly shows that the only studies specifically exploring the bibliometric data of *QI/SACJ* were conducted over a decade ago. Moreover, none of these studies covered the journal's entire history from its inception (1979) to 2023. Similarly, broader stud-

⁵ CiteScore: "CiteScore is based on the number of citations to documents (articles, reviews, conference papers, book chapters and data papers) in a journal over four years, divided by the number of the same document types indexed in Scopus and published in the same four years by that journal. CiteScore metrics are part of an evolving basket of metrics that will continue to grow with input and guidance from the research community." (Elsevier, 2018, p. 9)

⁶ SJR: "SCImago Journal Rank is a prestige metric, whose methodology is similar to that of Google PageRank. It weights the value of a citation depending on the field, quality and reputation of the journal that the citation comes from, so that 'all citations are not equal'. SJR also takes differences in the behavior of academics in different disciplines into account, and can be used to compare journals in different fields. The average SJR value for all journals in Scopus is 1.000." (Elsevier, 2018, p. 9)

⁷ SNIP: "Source-Normalized Impact per Paper is a ratio between the 'Raw Impact per Paper', a type of Citations per Publication calculation, actually received by the journal, compared to the 'Citation Potential', or expected Citations per Publication, of that journal's field. SNIP takes differences in disciplinary characteristics into account, and can be used to compare journals in different fields. The average SNIP value for all journals in Scopus is 1.000." (Elsevier, 2018, p. 9)

Table 1: Top 20 popular publication outlets for RSA authors in Computer Science

Publication name	Type	#Articles
<i>Lecture Notes in Computer Science</i> (Springer)	Proceedings	337
<i>South African Computer Journal</i> (SAICSIT)	Journal	201
<i>IEEE Access</i> (Institute of Electrical and Electronics Engineers)	Journal	161
<i>Communications in Computer and Information Science</i> (Springer)	Proceedings	132
<i>Advances in Intelligent Systems and Computing</i> (Springer)	Proceedings	76
<i>Scientometrics</i> (Springer)	Journal	59
<i>IFIP Advances in Information and Communication Technology</i> (Springer)	Proceedings	54
<i>Discrete Mathematics and Theoretical Computer Science</i> (La Maison des Mathématiques et de l'Informatique)	Journal	51
<i>Bioinformatics</i> (Oxford)	Journal	47
<i>Computers and Security</i> (Elsevier)	Journal	47
<i>Mathematical and Computer Modelling</i> (Elsevier)	Journal	39
<i>Computers and Chemical Engineering</i> (Elsevier)	Journal	34
<i>Electronic Journal of Information Systems Evaluation</i> (Academic Conferences International)	Journal	34
<i>Lecture Notes in Artificial Intelligence</i> (Springer)	Proceedings	34
<i>Theoretical Computer Science</i> (Elsevier)	Journal	33
<i>Computers and Education</i> (Elsevier)	Journal	30
<i>Journal of Combinatorial Optimization</i> (Springer)	Journal	30
<i>Journal of Molecular Modeling</i> (Springer)	Journal	29
<i>Lecture Notes in Business Information Processing</i> (Springer)	Proceedings	29
<i>Structural and Multidisciplinary Optimization</i> (Springer)	Journal	28

ies on local Computing research have failed to span this full period. This gap in knowledge highlights the need for a new, comprehensive and encompassing study focusing exclusively on the profile of *QI/SACJ*. Such an analysis would offer a deeper understanding of the journal's evolution and role in shaping the Computing field in South Africa, as well as a scientific basis for assessing its quality and impact. While previous *SACJ* studies explored topical/thematic, institutional and citation perspectives, this study seeks to update and expand on those findings by providing a thorough bibliometric and altmetric analysis covering the journal from 1979 to 2023.

5 METHODOLOGY

This study presents a bibliometric and altmetric profile of *SACJ* over 45 years (1979–2023), focusing on research articles. Letters to the editor, book reviews and announcements were excluded from the analysis. The data for this study were collected in October 2024 (see Supplements M and N in Naudé and Kroeze (2025)).

The bibliographic details for *QI* (1979–1989) and *SACJ* (1990–2009) were exported from the Inspec database in MS (Microsoft) Excel spreadsheet format. The bibliographic records for *SACJ* (2010–2023) were similarly exported from the Sabinet African Journals database, after which the data from Inspec and Sabinet were merged. To ensure accuracy, the retrieved bibliographic records were cross-checked against the *QI* and *SACJ* journal contents pages archived on the UIR (1979–2001) (SAICSIT, 1979–2021), the Sabinet database (2001–2008) (Sabinet, 2025b) and the *SACJ* websites (2009–2025) (*SACJ*, 2009–2023, 2021–2025).

Google Scholar, which covered both *SACJ* and *QI* throughout the 45-year analysis, was selected as the primary citation data source for this study. Google Scholar citations for *QI* and *SACJ* were retrieved using Harzing's Publish or Perish software (v 8.16) (Harzing, 2016). The journals' ISSN numbers were searched in Harzing, and the Google Scholar citation data were downloaded in MS Excel. The average number of *SACJ* citations per article per annum was calculated by dividing the citation count (number of citations) by scholarly productivity (number of papers) (Elsevier, 2018).

SACJ is not indexed by the Clarivate Web of Science database (Clarivate, 2025), but it has been indexed by Elsevier's citation-enhanced database Scopus (Elsevier, 2025) since 2016. The citation data for *SACJ* were downloaded from the database and exported to MS Excel. The data from 2016 to 2023 were compared to the Google Scholar citation data.

Mendeley readership counts were selected as alternative metric indicator for this study and were manually recorded in a spreadsheet for each article. Readership data were collected only for *SACJ* articles, as the *QI* articles were too dated (Mendeley was launched in 2008 and acquired by Elsevier in 2013 (Haunschild, 2020)). Mendeley readership is defined as “*the number of users who have saved documents to their private libraries [on the platform]. The more users that save a publication, the more [the] readership attributed to the publication*” (Zahedi & Costas, 2020). The average number of Mendeley readers per article was calculated by dividing the total number of readers by the total number of articles.

The number of authors for each *QI* and *SACJ* article was recorded in a spreadsheet to analyse the journal's authorship patterns and collaboration activities. The full counting method was applied, meaning each author received full credit for every publication s/he co-authored, regardless of the total number of co-authors (Perianes-Rodriguez et al., 2016). Each appearance of an author on a publication was counted as one full unit of authorship, independent of how many papers the author contributed to the journal. The average number of authors per article was calculated by dividing the total number of authors by the total number of articles.

The degree of collaboration (C) was calculated for *SACJ* for the period 1979–2023. Subramanyam defines (C) as “*the ratio of the number of collaborative research papers to the total*

number of research papers published in the discipline during a certain period of time” (Subramanyam, 1983). It is calculated by dividing the total number of multi-authored articles by the total number of articles.

6 RESULTS

6.1 Journal productivity

For the purpose of this study, journal productivity is defined as the number of articles published by *SACJ* per annum. Figures 1 to 3 present data from 1979 to 2023, specifically detailing the number of articles published each year. It also includes the Google Scholar citations, Scopus Citations and Mendeley readers within each year, as well as the average citations and readers (see Supplements O and P in Naudé and Kroeze (2025) for detailed tables).

In total, 768 articles were published in *QI* and *SACJ* between 1979 and 2023, with the journals’ productivity fluctuating significantly during this period. The average number of articles per year for *QI* and *SACJ* combined, was 17. The highest productivity was in 1999 with 51 articles, and the lowest was three in 1980.

The early years (1979–1989) generally displayed lower journal productivity. In *QI*, 131 articles were published between 1979 and 1989 (11 years), averaging approximately 11.9 articles per year. As shown in Figure 1, the highest productivity was in 1987 (27 articles). *QI* published no articles in 1981.

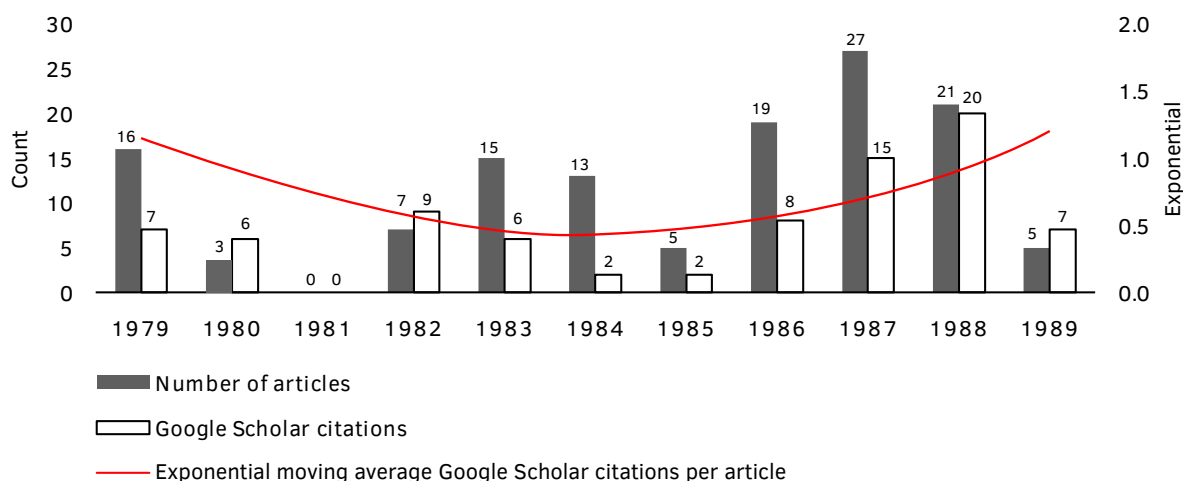


Figure 1: *QI* journal productivity and citations, 1979-1989

Figures 2 and 3 indicate that 637 *SACJ* articles were published between 1990 and 2023, spanning a period of 34 years. In *SACJ*, the majority of articles were published in 1999, with a peak of 51 articles, while the lowest number was recorded in 2011, with only four articles published (Figure 2). On average, *SACJ* published approximately 18.7 articles per year, over this period.

Figure 2 presents *SACJ*'s journal productivity, along with Google Scholar citations and Mendeley readership data, from 1990–2015. The graph shows a trend of exponential growth in the number of readers and citations per article, with an outlier of Google citations in 1999.

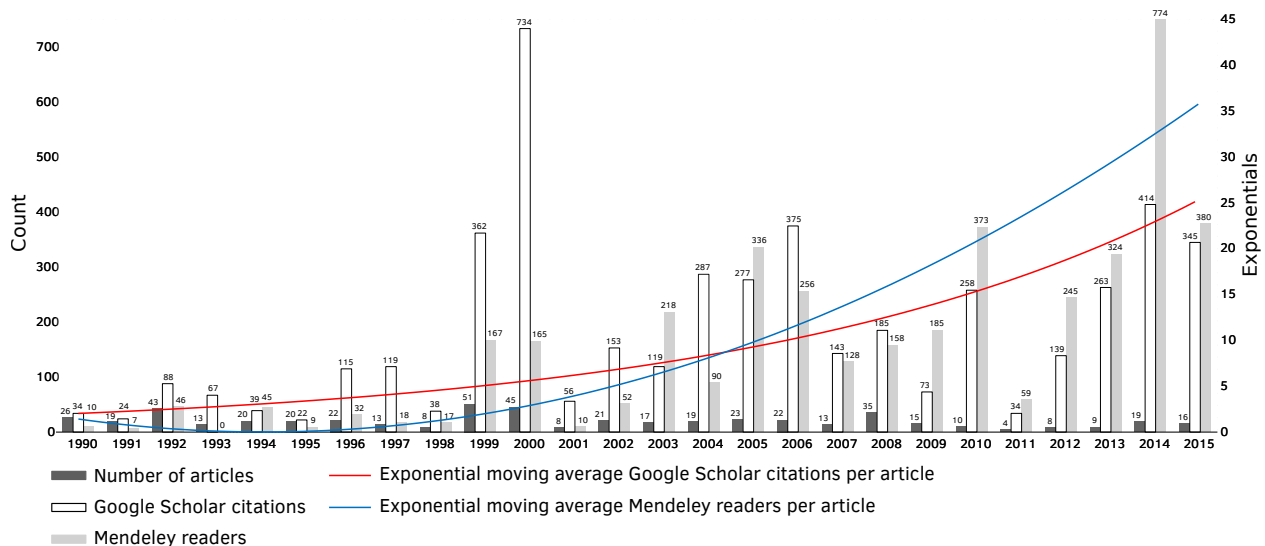


Figure 2: *SACJ* journal productivity, citations and Mendeley readers, 1990-2015

Figure 3 shows the *SACJ* journal productivity, Google Scholar citations, Scopus citations and Mendeley readers for the period 2016–2023. The number of articles published in this period peaked in 2017.

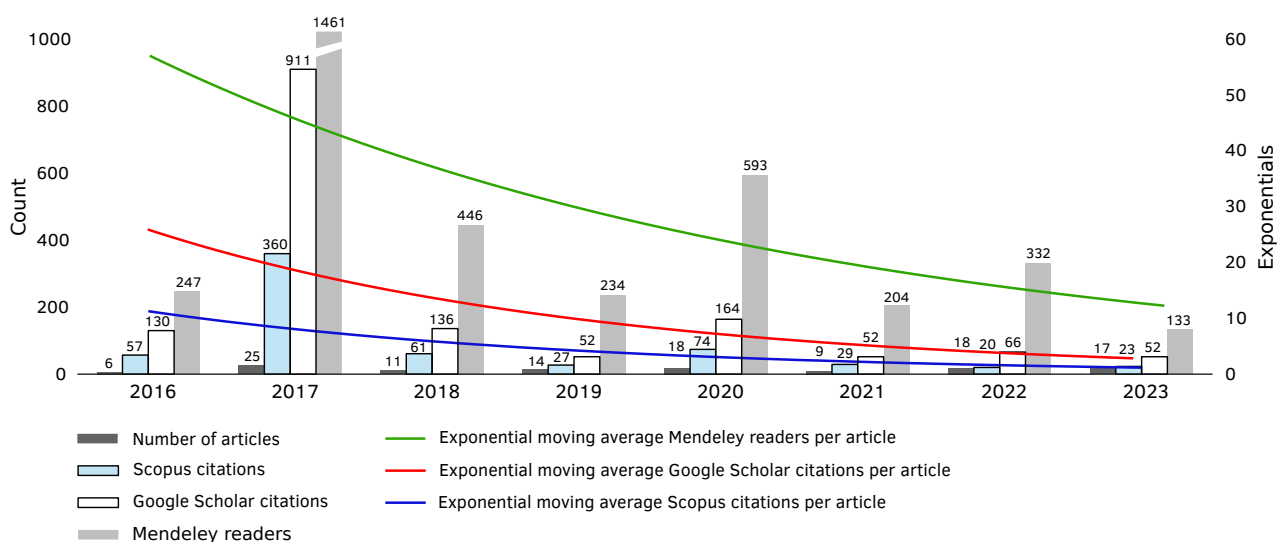


Figure 3: *SACJ* journal productivity, citations and Mendeley readers, 2016-2023

Prior to 2016, *SACJ* was numbered by issue, not volume. From 2016 onward, the journal adopted conventional volume numbering, beginning with volume 28 (see Supplement Q in Naudé and Kroeze (2025)).

6.2 Google Scholar and Scopus citations

Combined, *QI* and *SACJ* received a total of 6408 Google Scholar citations, with an overall average of 8.3 citations per article. Of the 768 articles published in both journals, 458 (60%) had citations, while the remaining 310 (40%) had none. Citation levels were low in the 1980s. As shown in Figure 1, *QI* received a total of 82 Google Scholar citations between 1979 and 1989, with an average of 0.6 citations per article. Of the 131 articles published in *QI*, only 26 (20%) received Google Scholar citations, while 105 (80%) received no citations.

Between 1990 and 2023, *SACJ* accumulated 6326 Google Scholar citations, averaging 9.9 citations per article (Figures 2 and 3). Of the 637 *SACJ* articles published during this period, 432 (68%) were cited in Google Scholar, while 205 (32%) remained uncited.

The average number of Google Scholar citations per article increased significantly after 2000, reflecting a clear upward trend (Figure 2). A comparison of three periods shows that the average Google Scholar citations per article rose from 3.9 (1990–1999) to 10.3 (2000–2009) and further to 20.7 (2010–2015). *SACJ* transitioned to an open-access journal in 2009 – a change that may have influenced citation rates.

Between 2016 and 2023, the 118 *SACJ* articles published received a total of 651 citations in Scopus, averaging 5.5 citations per article (Figure 3). Of these, 62 (52%) had Scopus citations while 56 (48%) had none. In comparison, Google Scholar data show a higher citation rate: 100 articles (85%) received citations, while only 18 (15%) remained uncited. Over the same period, 118 articles received 1563 Google Scholar citations, with an average of 13.2 citations per article.

The year 2017 marked the peak in scholarly impact, with the highest citation count recorded. Post-2017, as Figure 3 illustrates, there was a downward trend in citations and readers across all three platforms (Scopus, Google Scholar and Mendeley). Scopus citations dropped from an average of 14.4 per article in 2017 to 1.1 in 2022, while Google Scholar citations declined from 36.4 in 2017 to 3.1 in 2023. The average Google Scholar citations per article dropped from 23.5 (2016–2018) to 5.1 (2019–2023), indicating a downward trend in recent citation impact. This decline may partly reflect the shorter time newer articles have had to accumulate citations, underscoring the typical citation lag following their publication.

6.3 Mendeley readership

From 1990–2023, *SACJ* accumulated a total of 7754 Mendeley readers of 637 articles, resulting in an average of 12 per article (Figures 2 and 3). Of the 637 *SACJ* articles, 350 (55%) had Mendeley readers, while 287 (45%) had no readers.

Mendeley readership showed a significant upward trend, especially post-2000: a period comparison shows that the average number per article increased from 1.4 (1990–1999) to 7.3 (2000–2009) and then to 32.65 (2010–2015).

Figure 3 reveals that 2017 was the peak year, with the highest number of Mendeley readers (1461). However, post-2019 there was a noticeable decline, with the average number of Mendeley readers per article dropping from 58.4 in 2017 to 7.8 in 2023.

6.4 Authorship

Over the period 1979–2023, a total of 1464 authors published in *SACJ* and *QI* combined, with an average authorship rate of 1.9 authors per article (Tables 2 and 3). A total of 168 authors contributed to *QI*, with an average of 1.2 authors per article (Table 2). In comparison, *SACJ* had 1296 contributing authors and an average of 2.0 authors per article (Table 3).

Table 2: *QI*: Degree of collaboration (C) (1979-1999)

Year	Total articles	Single-authored articles	Multi-authored articles	Total authors	# Multi-authors	C
1979	16	13	3	21	8	0.19
1980	3	3	0	3	0	0.00
1981	0	0	0	0	0	0.00
1982	7	6	1	9	3	0.14
1983	15	11	4	20	9	0.27
1984	13	11	2	15	4	0.15
1985	5	4	1	7	3	0.20
1986	19	15	4	26	11	0.21
1987	27	21	6	33	12	0.22
1988	21	14	7	28	14	0.33
1989	5	4	1	6	2	0.20
Total	131	102	29	168	66	0.22

Table 2 shows that the degree of collaboration for *QI* is 0.22, indicating that 22% of the articles were multi-authored. *SACJ* shows a higher level of collaboration, with 0.70 (70% of articles) being authored by more than one author (Table 3). Combined, the overall degree of collaboration for *QI* and *SACJ* was 0.62, or 62%.

Table 2 shows that between 1979 and 1989, 131 articles were published in *QI*, of which 102 (78%) were single-authored and 29 (22%) were co-authored.

Of the 637 articles published in *SACJ* between 1990 and 2023, 193 (30%) were single-authored, while 444 (70%) were co-authored (Table 3).

When combining *QI* and *SACJ*, a total of 768 articles were published, with 295 (38%) being single-authored and 473 (62%) co-authored. This reflects a distinct contrast between the two journals: *QI* tended to favour single authorship, whereas *SACJ* demonstrated a stronger trend towards collaboration.

Figure 4 shows that authorship for the combined *QI* and *SACJ* dataset ranged from 1 to 7 authors. The highest productivity (i.e., number of articles) came from single-authored (293) and dual-authored (313) articles, comprising 79% of all articles. This was followed by three-authored articles (121) (16%).

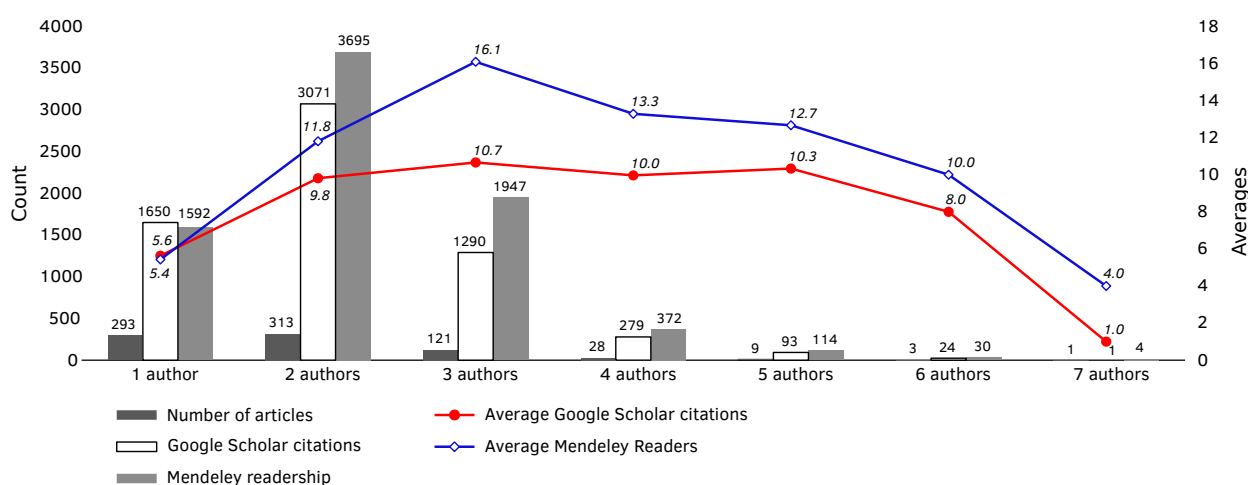


Figure 4: *QI* & *SACJ* authorship

The average Google Scholar citations increased from 5.6 (single author) to a peak of 10.7 (3 authors), suggesting that moderate co-authorship may enhance scholarly impact. After three authors, citation averages stabilised or declined slightly, notably to 10.3 (5 authors), 8 (6 authors), and only one citation for the single seven-author article (not significant, due to its singular occurrence). Mendeley readership data aligned with citation trends, peaking with three authors (16.1 readers per article), indicating that three-author articles were both well read and well cited.

Three-author articles achieved the highest average citations (10.7) and readership (16.1), while single-authored articles, despite being the most common (293 articles), had lower average citations (5.6) and readership (5.4). In contrast, while articles with four to six authors continued to show relatively strong averages, the data indicated diminishing returns in impact beyond three authors.

Table 3: *SACJ*: Degree of collaboration (C) (1990-2023)

Year	Total articles	Single-authored articles	Multi-authored articles	Total authors	# Multi-authors	C
1990	26	14	12	40	26	0.46
1991	19	6	13	37	31	0.68
1992	43	13	30	80	67	0.70
1993	13	5	8	24	19	0.62
1994	20	13	7	28	15	0.35
1995	20	10	10	33	23	0.50
1996	22	12	10	34	22	0.45
1997	13	5	8	23	18	0.62
1998	8	4	4	12	8	0.50
1999	51	18	33	102	84	0.65
2000	45	13	32	91	78	0.71
2001	8	2	6	18	16	0.75
2002	21	4	17	47	43	0.81
2003	17	5	12	32	27	0.71
2004	19	7	12	34	27	0.63
2005	23	9	14	45	36	0.61
2006	22	4	18	51	47	0.82
2007	13	2	11	28	26	0.85
2008	35	9	26	71	62	0.74
2009	15	3	12	40	37	0.80
2010	10	3	7	22	19	0.70
2011	4	0	4	11	11	1.00
2012	8	3	5	18	15	0.63
2013	9	2	7	21	19	0.78
2014	19	3	16	49	46	0.84
2015	16	5	11	33	28	0.69
2016	6	3	3	9	6	0.50
2017	25	4	21	58	54	0.84
2018	11	2	9	25	23	0.82
2019	14	2	12	31	29	0.86
2020	18	2	16	41	39	0.89
2021	9	0	9	23	23	1.00
2022	18	4	14	44	40	0.78
2023	17	2	15	41	39	0.88
Total	637	193	444	1296	1103	0.70

6.5 Influential articles

This section analyses the most influential articles based on citations and Mendeley readership. Table 4 presents the six most cited articles published in *QI* and *SACJ*, based on Google Scholar citation counts from 1979–2023.

Table 4: Influential articles: *QI* & *SACJ*, 1979–2023

Article	Google Scholar		Mendeley	
	Rank	Citations	Rank	Readers
van den Bergh and Engelbrecht (2000)	1	534	29	68
Sakpere et al. (2017)	2	277	2	275
Staudemeyer (2015)	3	238	18	97
Padayachee (2017)	4	194	4	187
Mawela et al. (2017)	5	134	3	217
Kortjan and von Solms (2014)	6	125	1	332

Kourie (2010) identified the article by van den Bergh and Engelbrecht (2000) as the most cited at the time (158 citations), and it remains the most cited *SACJ* article as at 2023, with 534 citations.

Table 4 highlights the four *SACJ* articles with the highest Mendeley readership. An article by Kortjan and von Solms (2014) recorded the highest number of Mendeley readers, despite ranking only sixth in terms of citation count. Table 4 suggests a partial overlap between the most cited articles and Mendeley readership in the journal.

6.6 Most productive authors

This section analyses the most active and leading contributors to *SACJ*. For the purpose of this study, a productive author is defined as having a publication frequency of nine or more articles in *QI* and *SACJ* between 1979 and 2023. Figure 5 presents a ranking of these 12 authors, based on their article productivity in *QI* and *SACJ* combined, followed by their Google Scholar citations and Mendeley reader counts. The most productive authors are P. Kotzé, D.G. Kourie, M.S. Olivier and I. Sanders.

Of the four productive authors identified by Kotzé and van der Merwe (2009) in the 2009 *SACJ* study, M.S. Olivier and J.H.P. Eloff remained among the authors with the most research articles in *SACJ*.

Of the productive authors, P. Kotzé (155) had the highest Google Scholar citation impact, followed by J.D. Roode (86) and M.S. Oliver (85). P. Kotzé also had the highest Mendeley readership (208), followed by A.J. van der Merwe (159) and A.P. Calitz (88).

There was no overlap between the authors of the most cited articles (Table 4) and the most productive authors (Figure 5), indicating that the most cited articles were not necessarily produced by the most productive authors, and vice versa.

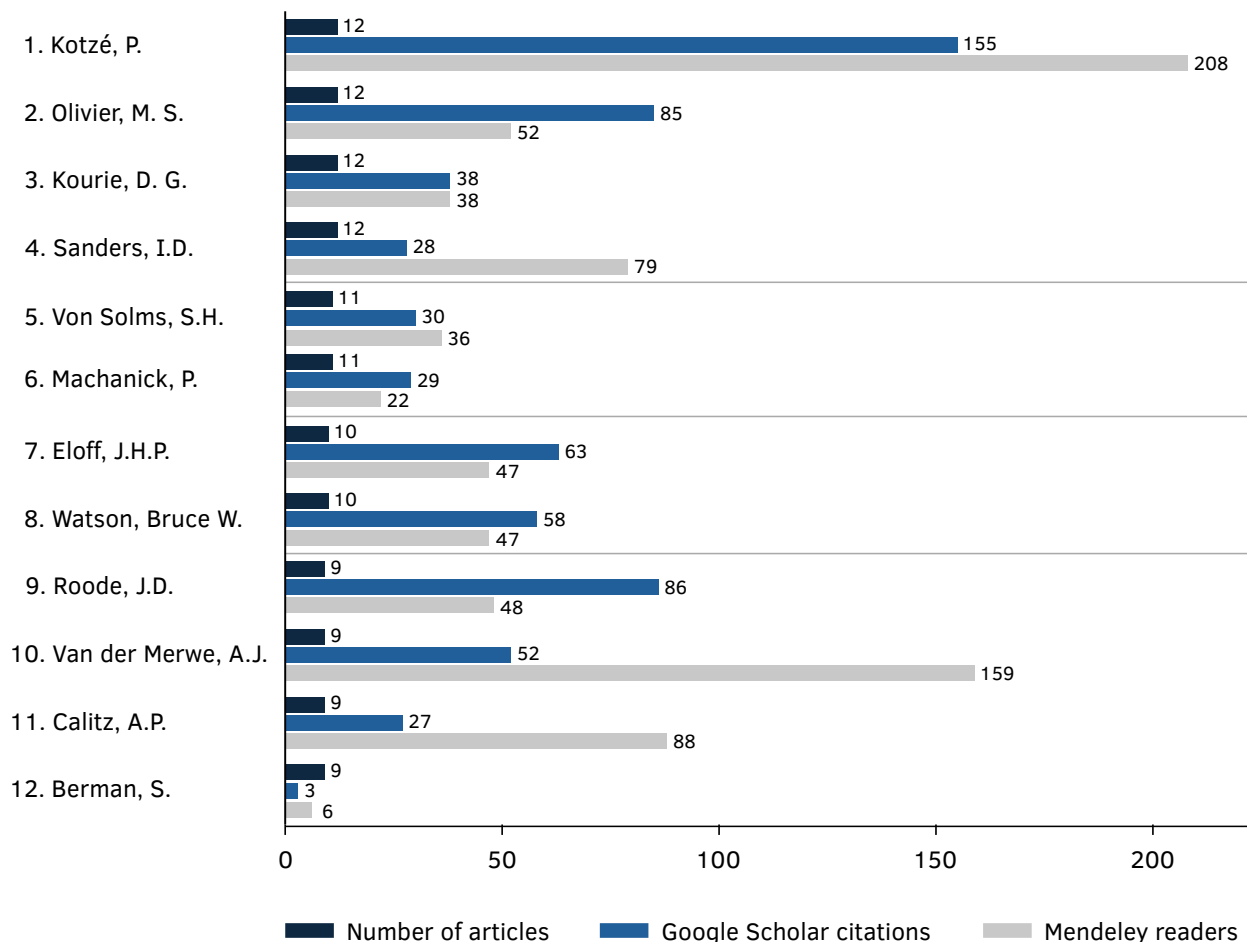


Figure 5: QI & SACJ's most productive authors, 1979-2023

6.7 Comparison of SACJ with other journals

SACJ is ranked in the Scopus Sources database (Elsevier, 2025) across several subject area categories as follows: 928th out of 1620 in *Education*, 335th out of 507 in *Computer Networks and Communications*, 322nd out of 474 in *Information Systems*, 669th out of 947 in *Computer Science Applications*, and 147th out of 186 in *Human–Computer Interaction*. SACJ is classified as a Quartile 3 journal in the Scopus database, indicating its position in the third tier of journals within its subject category, based on citation metrics and impact. Quartiles divide journals into four categories: Q1 (top 25%), Q2, Q3, and Q4 (bottom 25%). This indicates that SACJ (Q3) has a mid-to-lower-tier status as a journal.

Using the Scopus Sources database, SACJ was benchmarked against a selection of international journals in the Computing discipline. Given the vast number of journals, it was not feasible to include all comparable titles. Instead, a representative sample of Computer Science and Information Systems journals with a general scope was selected across different quartiles.

To ensure a fair comparison, a few regional and education-focused journals were included, acknowledging *SACJ*'s South African context and its strong performance within the education subset of the Scopus database.

Table 5 presents a benchmarking comparison using Scopus metrics, including CiteScore, citations (2020–2023), number of documents, percentage of cited articles, SNIP and SJR (SCImago Journal Rank). The journals range from elite, high-impact publications (e.g. *Proceedings of the IEEE*) to smaller regional or niche journals (e.g. *SACJ*, *Scandinavian Journal of Information Systems*). Journals that are regarded as educational are indicated with (E), and regional journals with (R).

Table 5: Comparison of *SACJ*'s journal metrics with other national and international journals

Source	Cite-Score	Citations (2020-2023)	Documents (2020-2023)	% Cited	SNIP	SJR
<i>Proceedings of the IEEE</i>	46.40	16286	351	93%	6.982	6.085
<i>European Journal of Information Systems</i>	23.07	3622	157	94%	3.766	3.824
<i>Artificial Intelligence Review</i>	21.97	20539	935	86%	5.068	3.260
<i>Journal of the Association for Information Systems</i>	11.17	2267	203	79%	2.384	2.302
<i>Proceedings of the ACM on Programming Languages</i>	5.16	4612	894	74%	1.876	1.242
<i>Communications of the Association for Information Systems</i>	3.93	1269	323	67%	0.893	0.620
<i>South African Journal of Science</i>	3.15	1138	361	65%	0.589	0.373
<i>Journal of Information Systems Education (E)</i>	2.75	330	120	62%	1.035	0.419
<i>Journal of Business Analytics</i>	2.49	132	53	66%	0.688	0.420
<i>Scandinavian Journal of Information Systems (R)</i>	1.96	94	48	67%	0.475	0.444
<i>South African Journal of Education (E)</i>	1.62	523	322	49%	0.691	0.292
<i>Foundations and Trends in Information Systems</i>	1.40	14	10	60%	0.421	0.186
<i>South African Computer Journal</i>	1.26	83	66	42%	0.293	0.213
<i>Systèmes d'Information et Management (R)</i>	1.16	37	32	56%	0.568	0.193
<i>AIS Transactions on Replication Research</i>	0.50	12	24	38%	0.272	0.241
<i>Journal of the Institute of Telecommunications Professionals</i>	0.06	6	97	6%	0.004	0.103

7 DISCUSSION

This single journal study offers a regional perspective on Computing research conducted in South Africa over the past 45 years. This section synthesises and reflects on the study's findings in relation to the journal's citation impact (via Google Scholar and Scopus), altmetric impact (via Mendeley readership), article production, patterns of authorship and collaboration.

Looking at the *SACJ* productivity and publication volume, three periods emerged:

Initial years (1979–1989): Publication numbers were relatively low and inconsistent, ranging from 0 (1981) to 27 (1987).

Growth period (1990–2000): Gradual increase in journal productivity, peaking at 51 articles in 1999.

Steady productivity (2001–2023): Publication numbers fluctuated between 4 and 35, indicating sustained productivity.

The initial years of *QI* showed low publication productivity. During the apartheid years, academic boycotts and sanctions were imposed on South African academic institutions, leading to the isolation of scholars. These measures had a detrimental effect, restricting international collaboration, influencing publication practices, and possibly reducing the citation impact of South African journals (Benatar, 1990).

There was a significant increase in journal productivity starting in the 1990s (towards the end of apartheid), with a notable peak in 1992. The highest overall number of articles appeared in 1999. The early 2000s also showed higher journal productivity. After 2000, the journal's productivity remained substantial, but did not reach the level of the late 1990s. More recent years (2018–2023) showed a slight decline in journal productivity.

An analysis of Google Scholar citation impact (Average Google Scholar citations per article) for *SACJ* revealed four phases:

Low-impact period (1979–1995): On average, each article received fewer than two citations.

Emerging influence (1996–2003): Multiple years exceeded five citations per article.

High-impact period (2004–2018): Many years showed double-digit average citations per article.

Decline (2019–2023): The average citations per article started to drop below ten from 2018 onward.

Between 1979 and 1989, 80% of the 131 *QI* articles remained uncited, which suggests that *QI* was still emerging and under-recognised. The journal may not have been widely indexed or accessible. The political isolation of South Africa during the apartheid era could have further hindered the visibility and impact of the journal. The 2000s to mid-2010s were the journal's most influential years.

Citations are a complex phenomenon influenced by numerous factors and variables that affect how often a paper is cited (Tahamtan et al., 2016). Below are some of the aspects that might contribute to the decline in *SACJ* citations from 2018 onwards.

This decline aligns with the broader global trend observed in the discipline of Computer Science, where the average number of citations per paper have declined over the past decade (Wahle et al., 2022), as reflected in the baseline citation rates (2015–2025) of the Clarivate Essentials Science Indicators (ESI), based on Web of Science data (Clarivate, 2025) (see Supplement R in Naudé and Kroeze (2025)). In ESI, field baseline citation rates represent the annualised average number of citations expected per paper within a specific research field.

The rapid growth in global research activity and publication volume (Cunningham & Smyth, 2025) has diluted the average citations per article. The more papers are published, the more the pool of available citations is spread across a larger body of papers, where more papers are competing for citations (Cavero et al., 2014). This expansion means that the average number of citations per paper may decline, because more publications are vying for scholarly attention (Garousi & Fernandes, 2017).

Changes in the scholarly publishing ecosystem, particularly the rise of open-access platforms and preprint servers, have contributed to a shift in citation patterns and diversion across multiple versions of the same work. Increasingly, researchers cite preprints and versions deposited on institutional repositories and academic social media sites, which can result in citations bypassing the final journal versions (Pagliaro, 2021).

Predatory journals may contribute to the distortion of citation patterns. The growing presence of those journals disrupts the citation landscape by diverting citations away from reputable, peer-reviewed journals. Inexperienced authors or those under pressure may inadvertently cite predatory sources instead of rigorous, reputable sources (Machanick, 2024). This misdirection not only reduces citations to legitimate journals, but also contaminates citation networks (Cunningham & Smyth, 2025). The inclusion of low-quality papers in citation-enhanced search engines like Google Scholar pollutes the scholarly system and has an impact on citation distribution.

Mendeley reader counts in this study closely align with traditional citation metrics from Google Scholar and Scopus (see Figures 2 and 3). A notable decline in Mendeley readership occurred after 2017, with the average number of readers per article decreasing from 58.4 in 2017 to just 7.8 in 2023. This downward trend may stem from the same factors which contributed to the decline in citation counts over the same period, as discussed above.

A result that demands more in-depth research is that there was no overlap between the authors of the most cited articles and the most productive authors. The literature indicates (Chang, 2021) that while highly cited authors are often associated with impactful and influential research, they are not necessarily the most productive in terms of publication volume. High-impact authors tend to focus on quality over quantity, producing fewer but more influential papers, compared to highly productive authors. Highly productive authors, by contrast, often publish a larger number of papers, but may not attain the same level of citation impact as highly cited authors. This suggests that productive authors prioritise quantity and may target journals with higher acceptance rates, but the relationship between productivity and citation impact is complex and influenced by many factors (Tahamtan et al., 2016). Further research could examine and compare the publication behaviour of highly cited versus highly productive South African Computing authors.

Over the 45 years analysed, the journal exhibited a clear shift from single- to multi-authorship. Between 1979 and 1989, only 22% of *QI* articles were co-authored, whereas from 1990 to 2023, 70% of *SACJ* articles were multi-authored. *SACJ* aligns with the global trend towards increased collaboration in Computer Science (Franceschet, 2011a), which has been accompanied by a steady rise in the average number of authors per paper over the recent

decades (Cavero et al., 2014; Fernandes & Monteiro, 2017; Kumari & Kumar, 2023). Several factors have contributed to this shift, notably the growing number of postgraduate students co-authoring publications with their supervisors (Brown & Tanner, 2008). Collaborative authorship has been shown to enhance both visibility and citation impact (Ibáñez et al., 2013). The increasing pressure to publish has also encouraged researchers to engage in collaboration as a means of distributing workload and boosting productivity (Cavero et al., 2014). The rise of interdisciplinary research also necessitated collaboration (Franceschet, 2011a).

The findings suggest a clear association between authorship patterns and scholarly impact, as measured by both Google Scholar citations and Mendeley readership. *SACJ* articles with two or three authors consistently outperformed others, indicating that moderate collaboration may enhance research visibility and academic influence. Notably, three-author papers achieved the highest average citations (10.7) and readership (16.1), while single-authored articles had the lowest averages in both metrics. Interestingly, while papers with four to six authors maintained relatively high averages, the marginal gains diminished, suggesting an optimal range of co-authorship. These results align with the existing literature, which suggests that collaboration can positively influence citation performance (Ibáñez et al., 2013). The analysis reinforced the value of collaborative authorship as a strategy for enhancing scholarly influence.

It is worth mentioning that the higher citation counts of multi-authored papers may partly result from self-citation in related or follow-up publications. Collaborative research tends to have higher self-citation rates than single-authored papers, with studies confirming that multi-authored papers receive higher self-citation rates (Leimu & Koricheva, 2005; Thelwall et al., 2023). This trend can be attributed to the broader network of co-authors, who are more likely to cite their own work. To provide a more accurate assessment, future analysis should consider excluding self-citations.

While *SACJ* still has room for improvement in terms of its CiteScore, it performs comparatively well when evaluated alongside similar regional and educational journals (see Table 5). (Although *SACJ* is not, per se, an educational journal, it covers educational technology and the teaching of Computing in South Africa as a key area (Kotzé & van der Merwe, 2009).) Most of these journals have a lower CiteScore (below 3), as does *SACJ*, than the top journals with scores ranging from 3 to 46. However, this premise should be researched in more detail in future work, since a comprehensive comparison falls outside the scope of this article. *SACJ* currently ranks in the lower tier of international Computing and Information Systems journals, based on Scopus journal metrics. However, it again compares reasonably well with regional and educational journals, especially considering its national focus and limited resources.

With reference to the comparison of *SACJ* and *LNCS*, it should be noted that with a smaller number of papers, the citation scores of *SACJ* are inherently more variable from year to year than those of a journal like *LNCS* that publishes a much higher volume of papers tending to render more stable metrics. For example, one outlier in *LNCS* will not make as big a difference as it would in *SACJ*. Moreover, the author scope of *SACJ* is focused on South African authors, while *LNCS* attracts papers on a global scale, which may also have an effect on the number

of citations drawn. It may, therefore, be considered as biased to compare these journals' cite scores. In this regard, Franceschet's (2011b) study is particularly noteworthy. He examined the asymmetry or skewness of citation distributions in Computer Science publications, considering both journal and conference papers, and acknowledging the field's distinctive reliance on conferences. The study revealed that citation distributions are highly asymmetric, with a small fraction of papers attracting a disproportionately large number of citations. Moreover, it found that conference papers display greater skewness in their citation patterns than journal papers, while journal articles, overall, tend to achieve a higher bibliometric impact.

The relatively low impact of *SACJ*, compared to well-known and highly cited international journals, may be due to the perception that local journals are of lower quality – see, for example, a concern raised by Hamlall and Belle (2019) regarding the disappointing productivity of South African researchers in Computer Science and Information Systems: *“Other areas of concern are that there is a failure among academics to distinguish between high-quality international journal publications and lower-quality locally accredited journals, which affects their incentive to publish.”* However, *SACJ*'s listing on Scopus (Elsevier, 2025), DOAJ (DOAJ, 2025) and SciELO SA (SciELO, 2025) supports its credibility and visibility, indicating that such a perception could be unfair (Machanick, 2024).

Collaboration is shaped by a complex mixture of individual, institutional, national and global factors. The DHET funding of tertiary institutions per research output (DHET, 2015) plays a significant role in shaping the collaboration and publication behaviour of South African researchers. The importance of a publication profile, NRF rating and promotion stimulates research on the one hand, but the incentives paid to individual researchers may discourage collaboration on the other hand (Sooryamoorthy, 2019).

South African universities increasingly encourage researchers to collaborate with international scholars and publish in high-impact journals to enhance institutional citation rates and global rankings (von Solms & von Solms, 2016). However, the DHET subsidy framework significantly influences researchers' choice of collaboration partners and publication outlets. Since researchers are incentivised to publish in DHET-accredited journals to secure government subsidies, this can create a tension: international collaborators often prefer publishing in high-impact journals, which may not always appear on the DHET-accredited list. Additionally, South African researchers may find that the regional focus of journals like *SACJ* limits the appeal for international co-authors. Nevertheless, increasing the number of internationally co-authored articles in *SACJ* could help to enhance the journal's citation impact and global visibility.

According to Hamlall and Belle (2019), research collaboration enhances both the visibility and citation rates of scholarly works. However, the funding model used to subsidise South African universities – partly based on research output – can inadvertently discourage collaboration. Under the DHET research output system, subsidies for co-authored publications are divided among the contributing institutions. This distribution reduces the financial benefit for any single institution, potentially disincentivising inter-institutional collaboration. In contrast, intra-institutional collaboration yields greater subsidy returns for universities, as the

full benefit remains within the same institution. Studies by Turpin (2018), van Biljon and Naude (2018), and Parry (2019) confirm a prevailing trend toward intra- rather than inter-institutional collaboration.

Moreover, following the lifting of sanctions against South Africa, international collaboration became increasingly feasible. Efforts to build a unified democracy amid complex social and historical challenges likely attracted significant interest from international scholars. This, combined with the global trend toward increased research collaboration, may help to explain the rise in collaboration.

One possible reason why collaboration leads to higher research throughput is that it enables the sharing of data, knowledge and expertise among researchers. Additionally, funding agencies often require tangible outputs from collaborative teams, and joint projects tend to enhance the visibility of research publications. Networking through collaboration also benefits emerging researchers by increasing their chances of receiving invitations to co-author publications and participate in further projects (Sooryamoorthy, 2014). Factors such as holding a PhD, years of academic experience, and the breadth of collaboration networks all influence a researcher's productivity (Sooryamoorthy, 2014). These findings suggest several promising directions for further empirical research (both qualitative and quantitative) such as conducting surveys and interviews with scholars, to gather in-depth insights and statistically test new hypotheses.

Since the correlation between collaboration and productivity varies in respect of subjects and countries (Sooryamoorthy, 2014), it was important to gain a better understanding of the status quo in the South African Computing guild. In this regard, the Computing scenario in South Africa needs to be researched in greater depth in future work.

8 CONCLUSION

This study offered a comprehensive overview of *SACJ* as a publication, and its citation trends over several decades. *SACJ* has played a pivotal role in developing and disseminating computing knowledge in South Africa. With a rich history and heritage spanning almost half a century, *SACJ* provided a vital platform for local researchers to publish their work, enhancing their visibility and citation impact. *SACJ* has fostered collaboration among South African researchers, and facilitated connections with international scholars, thereby significantly serving to advance Computing Science in this country.

In follow-up research, the citation rate of *SACJ* could be compared to that of South African conference proceedings such as SACLA and SAICSIT. While some of these proceedings are freely available on open-access platforms, others are hosted behind paywalls, including those of the ACM Digital Library and Springer.

Future work could also explore usage indicators such as downloads, views or social media indicators as other alternative metrics. Furthermore, future researchers could extend this study by using new and emerging artificial intelligence-powered analytical tools to identify and analyse research and collaboration trends.

9 ACKNOWLEDGEMENTS

The authors would like to thank (in no particular order) Derrick Kourie, Judith Bishop, Tendani Mawela, Katherine Malan, Ruth de Villiers, André Calitz, Charlotte Hlengwa, Tawanda Madavo, Philip Machanick, Ian Sanders, Paula Kotzé, Hussein Suleman, Darelle van Greunen, Brenda Scholtz, Patrick Marais, Stefan Gruner, Lerine Steenkamp, Basie von Solms, Rossouw von Solms, Martin Olivier, AURONA Gerber, Lisa Seymour and Marié Hattingh for their assistance in finding information and copies of old proceedings and journal issues; Juanita Beytell for her help with the figures; the following librarians who accepted donations of hard copies, CDs and memory sticks: Nestus Venter (NWU), Bulelwa Mandubu (UP), Nobuhle Maimela (UFS) and Nuroo Davids (UCT); Ansie van der Westhuizen, who created the SAICSIT archive on the UIR; Suezette Opperman for indexing the scanned copies; Sabinet (Thabang Methi and Tebogo Methi) for scanning the old printed proceedings and journal issues; the SAICSIT Council for funding the costs of digitising the old printed copies of proceedings and journal issues. We sincerely apologise if we have inadvertently omitted anyone by name. Over the course of more than seven years, we consulted with many individuals, and to each and every one of you, we extend our heartfelt thanks.

References

- Abramo, G., D'Angelo, C. A., & Reale, E. (2019). Peer review versus bibliometrics: Which method better predicts the scholarly impact of publications? *Scientometrics*, 121(1), 537–554. <https://doi.org/10.1007/S11192-019-03184-Y>
- ACM. (2025). Association for Computing Machinery [Online]. <https://dl.acm.org/>
- ACM-CCS. (2025). ACM Computing Classification System [Online]. <https://dl.acm.org/ccs>
- Araujo, A. C., Vanin, A. A., Nascimento, D. P., Gonzalez, G. Z., & Costa, L. O. P. (2021). What are the variables associated with altmetric scores? *Systematic reviews*, 10(1), 193. <https://doi.org/10.1186/S13643-021-01735-0>
- ASSAf. (2019). *Report on grouped peer review of scholarly journals in communication and information sciences* (tech. rep.). Academy of Science of South Africa. <https://doi.org/10.17159/ASSAF.2019/0041>
- ASSAf. (2025). Khulisa journals [Accessed 23 September 2025]. <https://journals.assaf.org.za/>
- Benatar, S. R. (1990). An alternative to academic boycott. *Nature*, 343(6258), 505–506. <https://doi.org/10.1038/343505a0>
- Brown, I., & Tanner, M. (2008). The international visibility of South African IS research: An author-affiliation analysis in the top-ranked IS-centric journals. *South African Computer Journal*, 41, 14–20. <https://journals.co.za/doi/10.10520/EJC28079>
- Calitz, A. P. (2022). The 50 year history of SACL and computer science departments in South Africa. *Communications in Computer and Information Science*, 1461 CCIS, 3–23. https://doi.org/10.1007/978-3-030-95003-3_1

- Cavero, J. M., Vela, B., & Cáceres, P. (2014). Computer science research: More production, less productivity. *Scientometrics*, 98(3), 2103–2111. <https://doi.org/10.1007/S11192-013-1178-2>
- Chang, Y. W. (2021). Characteristics of high research performance authors in the field of library and information science and those of their articles. *Scientometrics*, 126(4), 3373–3391. <https://doi.org/10.1007/S11192-021-03898-Y>
- Clarivate. (2025). Web of science database [Accessed 23 September 2025]. <https://www.webofscience.com/>
- Cunningham, P., & Smyth, B. (2025). An analysis of the impact of gold open access publications in computer science. *Communications of the ACM*, 68(8), 62–69. <https://doi.org/10.1145/3721975>
- DBLP. (2025). DBLP computer science bibliography [Online]. <https://dblp.org/>
- DHET. (2015). Research outputs policy [Accessed 23 September 2025]. *Government Gazette*, 597(38552), 3–31. <https://www.dhet.gov.za/Policy%20and%20Development%20Support/Research%20Outputs%20Policy%202015.pdf>
- DHET. (2025). *DHET list of accredited journals* [Accessed 17 October 2025]. <https://www.dhet.gov.za/SitePages/University%20Research%20Support%20and%20Policy%20Development.aspx>
- Dimensions. (2025). Dimensions: A digital science solution [Accessed 23 September 2025]. <https://www.dimensions.ai/>
- DOAJ. (2025). Directory of open access journals [Accessed 23 September 2025]. <https://doaj.org/>
- Donthu, N., Kumar, S., Mukherjee, D., Pandey, N., & Lim, W. M. (2021). How to conduct a bibliometric analysis: An overview and guidelines. *Journal of Business Research*, 133, 285–296. <https://doi.org/10.1016/J.JBUSRES.2021.04.070>
- Elsevier. (2025). Scopus database [Accessed 23 September 2025]. <https://www.scopus.com/>
- Elsevier. (2018). *Research metrics guidebook*. <https://www.elsevier.com/en-in/products/scopus/metrics>
- Fernandes, J. M., & Monteiro, M. P. (2017). Evolution in the number of authors of computer science publications. *Scientometrics*, 110(2), 529–539. <https://doi.org/10.1007/S11192-016-2214-9>
- Franceschet, M. (2011a). Collaboration in computer science: A network science approach. *Journal of the American Society for Information Science and Technology*, 62(10), 1992–2012. <https://doi.org/10.1002/ASI.21614>
- Franceschet, M. (2011b). The skewness of computer science. *Information Processing & Management*, 47(1), 117–124. <https://doi.org/10.1016/J.IPM.2010.03.003>
- García-Villar, C. (2021). A critical review on altmetrics: Can we measure the social impact factor? *Insights into Imaging*, 12, 1–10. <https://doi.org/10.1186/S13244-021-01033-2>

- Garousi, V., & Fernandes, J. M. (2017). Quantity versus impact of software engineering papers: A quantitative study. *Scientometrics*, 112(2), 963–1006. <https://doi.org/10.1007/S11192-017-2419-6>
- Gerber, A. (Ed.). (2022). *SAICSIT 2022: Proceedings of 43rd conference of the South African Institute of Computer Scientists and Information Technologists*. EPiC Series in Computing. https://easychair.org/publications/volume/SAICSIT_2022
- Gerber, A. (Ed.). (2024a). *SAICSIT'24 – Online proceedings of the South African Institute of Computer Scientists and Information Technologists 2024 conference: Human-machine-digital-convergence*. SAICSIT. <https://saicsit2024.mandela.ac.za/saicsit2024/media/Store/documents/SAICSITOnline.pdf>
- Gerber, A. (Ed.). (2024b). *South African Computer Science and Information Systems Research Trends, 45th Annual conference proceedings*. SAICSIT 2024. <https://doi.org/10.1007/978-3-031-64881-6>
- Gerber, A. (Ed.). (2025a). *Proceedings of the 46th annual conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT 2025)*. SAICSIT. <https://saicsit2025.org.za/online-main.pdf>
- Gerber, A. (Ed.). (2025b). *South African computer science and information systems research trends, 46th annual conference, SAICSIT 2025* (Vol. 2583). Springer Nature. <https://doi.org/10.1007/978-3-031-96262-2>
- Gerber, A., & Coetzee, M. (Eds.). (2023a). *South African Institute of Computer Scientists and Information Technologists – 44th Annual conference (SAICSIT 2023)* (Vol. 1878). Springer Nature Switzerland. <https://doi.org/10.1007/978-3-031-39652-6>
- Gerber, A., & Coetzee, M. (Eds.). (2023b). *SAICSIT 2023 Proceedings of the 44th South African Institute of Computer Scientists and Information Technologists Conference: Connect to be the future*. <https://saicsit2023.org/proceedings>
- Hamlall, D., & Belle, J. P. V. (2019). Research barriers experienced by South African academics in information systems and computer science. *Communications in Computer and Information Science*, 963, 95–107. https://doi.org/10.1007/978-3-030-05813-5_7
- Harzing, A.-W. (2016). Publish or perish [Accessed 23 September 2025]. <https://harzing.com/resources/publish-or-perish>
- Haunschild, R. (2020). Mendeley. *Handbook Bibliometrics*, 281–287. <https://doi.org/10.1515/9783110646610-028>
- Ibáñez, A., Bielza, C., & Larrañaga, P. (2013). Relationship among research collaboration, number of documents and number of citations: A case study in Spanish computer science production in 2000–2009. *Scientometrics*, 95(2), 689–716. <https://doi.org/10.1007/S11192-012-0883-6>
- IET. (2025). Institution of Engineering and Technology (IET) – Inspec database [Accessed 23 September 2025]. <https://www.theiet.org/publishing/inspec>
- IFIP. (2025). International Federation for Information Processing [Online]. <https://www.ifip.org/>

- IITPSA. (2025). Institute of Information Technology Professionals South Africa [Online]. <http://www.iitpsa.org.za/>
- Kortjan, N., & von Solms, R. (2014). A conceptual framework for cyber-security awareness and education in SA: Research article. *South African Computer Journal*, 52(1), 29–41. <https://journals.co.za/content/comp/52/1/EJC154952>
- Kotzé, P., & van der Merwe, A. (2009). The research foci of computing research in South Africa as reflected by publications in the South African Computer Journal. *South African Computer Journal*, 44, 67–84. <https://doi.org/10.18489/SACJ.V44I0.24>
- Kourie, D. (1989). Editorial. *Quaestiones Informaticae*, 6(4), 137–138. <https://doi.org/10.6084/m9.figshare.12728294>
- Kourie, D. (2010). The South African Computer Journal: 1989 to 2010. *Transactions of the Royal Society of South Africa*, 65(2), 107–111. <https://doi.org/10.1080/0035919X.2010.510664>
- Kumari, P., & Kumar, R. (2023). Collaborative authorship patterns in computer science publications. *Annals of Library and Information Studies*, 70(1), 22–32. <https://doi.org/10.56042/alis.v70i1.70536>
- Leimu, R., & Koricheva, J. (2005). Does scientific collaboration increase the impact of ecological articles? *BioScience*, 55(5), 438–443. [https://doi.org/10.1641/0006-3568\(2005\)055\[0438:DSCITI\]2.0.CO;2](https://doi.org/10.1641/0006-3568(2005)055[0438:DSCITI]2.0.CO;2)
- Machanick, P. (2024). Mentorship lessons from growing a developing country journal. *Communications of the ACM*, 67(11), 35–37. <https://doi.org/10.1145/3663472>
- Malan, K. M. (2023). Editorial: A new era for SACJ. *South African Computer Journal*, 35(2), vii–ix. <https://doi.org/10.18489/SACJ.V35I2.17439>
- Mawela, T., Ochara, N. M., & Twinomurinzi, H. (2017). e-Government implementation: A reflection on South African municipalities. *South African Computer Journal*, 29(1), 147–171. <https://doi.org/10.18489/SACJ.V29I1.444>
- Mouton, J., J. Blanckenberg, M., van Lill & Redelinghuys, H. (2022). A scientometric assessment of the computer sciences in South Africa, Keynote presentation at the 51st annual conference of the southern african computer lecturers' association [Available online]. <https://doi.org/10.6084/m9.figshare.12728294>
- Mouton, J., Basson, I., Treptow, R., & Blanckenberg, J. (2019). The state of the South African research enterprise [Accessed 23 September 2025]. https://www.researchgate.net/publication/335857684_The_state_of_the_South_African_research_enterprise
- Naudé, F., & Kroeze, J. H. (2024). South African research contributions to Lecture Notes in Computer Science, 1973-2022. *South African Journal of Science*, 120(27–40). <https://doi.org/10.17159/SAJS.2024/15199>
- Naudé, F., & Kroeze, J. H. (2025). South African Computer Journal (1979–2023) [Dataset]. <https://doi.org/10.6084/m9.figshare.12728294>
- Padayachee, K. (2017). A snapshot survey of ICT integration in South African schools. *South African Computer Journal*, 29(2), 36–65. <https://doi.org/10.18489/SACJ.V29I2.463>

- Pagliaro, M. (2021). Did you ask for citations? An insight into preprint citations en route to open science. *Publications*, 9(3), 26. <https://doi.org/10.3390/PUBLICATIONS9030026>
- Parry, D. A. (2019). Computing research in South Africa: A scientometric investigation. *South African Computer Journal*, 31(1), 51–79. <https://doi.org/10.18489/SACJ.V31I1.674>
- Perianes-Rodriguez, A., Waltman, L., & van Eck, N. J. (2016). Constructing bibliometric networks: A comparison between full and fractional counting. *Journal of Informetrics*, 10(4), 1178–1195. <https://doi.org/10.1016/J.JOI.2016.10.006>
- Sabinet. (2025a). Accredited journals [Accessed 23 September 2025]. <https://journals.co.za/publications-accredited>
- Sabinet. (2025b). South African Institute of Computer Scientists and Information Technologists, *South African Computer Journal* [Accessed 23 September 2025]. <https://journals.co.za/journal/comp>
- SACJ. (2009–2023). South African Computer Journal Archives [Accessed 23 September 2025]. <https://sacj.cs.uct.ac.za/index.php/sacj/issue/archive>
- SACJ. (2021–2025). South African Computer Journal Archives [Accessed 23 September 2025]. <https://sacj.org.za/issue/archive>
- SACLA. (2025). South African Institute of Computer Scientists and Information Technologists [Accessed 23 September 2025]. <http://www.sacla.org.za/>
- SAICSIT. (2025). South African Institute of Computer Scientists and Information Technologists [Accessed 23 September 2025]. <http://saicsit.org>
- SAICSIT. (1979–2021). SAICSIT digital archive [Accessed 29 September 2025]. <https://uir.unisa.ac.za/communities/5e256de7-575c-4e5e-bc6b-2e9a7907198c>
- Sakpere, W., Oshin, M. A., & Mlitwa, N. B. (2017). A state-of-the-art survey of indoor positioning and navigation systems and technologies. *South African Computer Journal*, 29(3), 145–197. <https://doi.org/10.18489/SACJ.V29I3.452>
- Sanders, I. D., & Alexander, P. M. (2015). A study of computing doctorates in South Africa from 1978 to 2014. *South African Computer Journal*, 57, 58–89. <https://doi.org/10.18489/SACJ.V0I57.294>
- SciELO. (2025). SciELO – Scientific Electronic Library Online [Accessed 23 September 2025]. <https://www.scielo.org.za/>
- Singh, S., Mujinga, M., Lotriet, H., & Tait, B. (Eds.). (2021, September). *SAICSIT Conference 2021: Proceedings of the South African Institute of Computer Scientists and Information Technologists*. Unisa Press. <https://hdl.handle.net/10500/28972>
- Sooryamoorthy, R. (2014). Publication productivity and collaboration of researchers in South Africa: New empirical evidence. *Scientometrics*, 98(1), 531–545. <https://doi.org/10.1007/S11192-013-0990-Z>
- Sooryamoorthy, R. (2019). Scientific knowledge in South Africa: Information trends, patterns and collaboration. *Scientometrics*, 119(3), 1365–1386. <https://doi.org/10.1007/S11192-019-03096-X>

- Staudemeyer, R. C. (2015). Applying long short-term memory recurrent neural networks to intrusion detection. *South African Computer Journal*, 56(1), 136–154. <https://doi.org/10.18489/SACJ.V56I1.248>
- Subramanyam, K. (1983). Bibliometric studies of research collaboration: A review. *Journal of Information Science*, 6(1), 33–38. <https://doi.org/10.1177/016555158300600105>
- Tahamtan, I., Afshar, A. S., & Ahamdzadeh, K. (2016). Factors affecting number of citations: A comprehensive review of the literature. *Scientometrics*, 107(3), 1195–1225. <https://doi.org/10.1007/S11192-016-1889-2>
- Thelwall, M. (2020). The pros and cons of the use of altmetrics in research assessment. *Scholarly Assessment Reports*, 2(1), 1–9. <https://doi.org/10.29024/SAR.10>
- Thelwall, M., Kousha, K., Abdoli, M., Stuart, E., Makita, M., Wilson, P., & Levitt, J. (2023). Why are coauthored academic articles more cited: Higher quality or larger audience? *Journal of the Association for Information Science and Technology*, 74(7), 791–810. <https://doi.org/10.1002/ASI.24755>
- Turpin, M. (2018). Assessing South African ICT4D research outputs: A journal review. *South African Computer Journal*, 30(1), 108–127. <https://doi.org/10.18489/SACJ.V30I1.541>
- van Biljon, J., & Naude, F. (2018). Collaboration towards a more inclusive society: The case of South African ICT4D researchers. *IFIP Advances in Information and Communication Technology*, 537, 82–94. https://doi.org/10.1007/978-3-319-99605-9_6
- van den Bergh, F., & Engelbrecht, A. (2000). Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26, 84–90. <https://hdl.handle.net/10520/EJC27898>
- von Solms, R., & von Solms, B. (2016). Publish or perish — But where? *South African Computer Journal*, 28(1), 44–58. <https://doi.org/10.18489/SACJ.V28I1.394>
- Wahle, J. P., Ruas, T., Mohammad, S. M., & Gipp, B. (2022). D3: A massive dataset of scholarly metadata for analyzing the state of computer science research. *2022 Language Resources and Evaluation Conference, LREC 2022*, 2642–2651. <https://arxiv.org/abs/2204.13384v4>
- Zahedi, Z., & Costas, R. (2020). Do online readerships offer useful assessment tools? Discussion around the practical applications of Mendeley readership for scholarly assessment. *Scholarly Assessment Reports*, 2(1), 1–14. <https://doi.org/10.29024/SAR.20>

Investigating the support of proprioception and visual guidance for menu selection in virtual reality

Kwan Sui Dave Ka , Isak de Villiers Bosman , Theo J.D. Bothma 

University of Pretoria, PO Box 14679, Hatfield, 0028, Pretoria, South Africa

ABSTRACT

Virtual reality (VR) technology enhances spatial awareness in the virtual environment, yet current menu design research seldom explores how this can support interaction. This study investigates how proprioception, users' sense of body position, can guide menu selection in immersive VR environments. A menu system was developed to test how users interact with menu items using non-visual senses, supported by visual cues. Across three usability sessions, participants engaged with the system, and their evolving behaviours were examined through performance metrics, observations, interviews, and focus groups. Thematic analysis revealed that while users can learn to use proprioceptive cues for menu selection, they rarely rely on them without visual guidance. Effective use requires systems that support spatial awareness and familiarity, allowing interactions to become intuitive and memorised over time. The study also identifies types of visual guidance linked to varying levels of familiarity with surrounding virtual objects. These findings contribute to VR interaction design by demonstrating how visual and proprioceptive cues affect user behaviour and support efficient, embodied menu selection.

Keywords virtual reality, menu system, visual guidance, proprioception, user behaviour

Categories • Human-centred interaction ~ Human computer interaction, Interaction paradigms, Virtual reality

Email

Kwan Sui Dave Ka – dave.ka@up.ac.za (CORRESPONDING)
Isak de Villiers Bosman – isak.bosman@up.ac.za
Theo J.D. Bothma – theo.bothma@up.ac.za

Article history

Received: 30 July 2023
Accepted: 13 Jun 2025
Online: 22 December 2025

1 INTRODUCTION

Virtual reality (VR) technology can provide an immersive experience for users to feel part of a three-dimensional (3D) virtual environment, making an accessible way for an embodied and personal experience that may otherwise be challenging, e.g., viewing Mount Everest from the summit (Sólfar Studios, 2017), or impossible, e.g., inspecting enlarged visualisations of small animal organs (Liimatainen et al., 2021). These kinds of experiences need different functions to help users interact with virtual elements in these virtual environments.

Ka, K.S.D., Bosman, I. de V., Bothma, T.J.D. (2025). Investigating the support of proprioception and visual guidance for menu selection in virtual reality . *South African Computer Journal* 37(2), 37–73. <https://doi.org/10.18489/sacj.v37i2.21524>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](https://creativecommons.org/licenses/by-nc/4.0/) 
SACJ is a publication of [SAICSIT](https://www.sacj.org/). ISSN 1015-7999 (print) ISSN 2313-7835 (online)

A method for doing so is to make use of a menu system which serves as a navigational tool that allows users to access functions and features systematically, ensuring that the interaction with a digital environment is intuitive and productive (Bailly et al., 2016). Well-designed menu systems facilitate clear pathways to functionalities, thereby reducing cognitive load and easing learning curves that can accommodate both novice and expert users through structured layouts and familiar design patterns (Shneiderman et al., 2018). By integrating principles of usability and user-centred design, menu systems help streamline tasks while also enhancing overall user satisfaction, making them a crucial aspect of effective user-interface (UI) design. There is substantial literature that provides usability guidelines and principles

There is substantial literature that provides usability guidelines and principles for menu system interactions (Bowman & Wingrave, 2001; Cockburn et al., 2007; Eriksson, 2016). Existing research on menu system design in 3D virtual environments has emphasised the importance of enhancing immersion. This is achieved by making menu systems visually believable as part of the virtual environment, i.e., diegetic (Bailly et al., 2016), enabling more natural interaction and improving the overall user experience.

Building upon this idea of improving menu systems design by supporting an immersive experience, the intended outcome of this study was to explore interactions with menu systems that feel natural and intuitive. Therefore, a more immersive approach was investigated to support the selection process. Improving the usability of the selection process for menu systems is important because this action is done repeatedly and rapidly (Argelaguet & Andujar, 2013).

In addition to the visual representation of menu systems, there has been research conducted to optimise selection by investigating the impact of the layout for menu items (Bailly et al., 2016). However, this does not include investigating the improvement of menu item selection, specifically regarding the action of selection for menu systems within the context of immersive technologies like VR. Virtual reality interactions have also been thoroughly explored, some of which are used to inform interaction design for menu systems in VR. In VR, where the user feels part of the virtual space, this feeling can be investigated as a way to consider the use of spatial awareness of menu items in the virtual space.

Since VR technology can provide an embodied experience, it presents an opportunity to leverage spatial awareness within the virtual environment to support more intuitive and effective menu item selection. Visual, audio, and haptic feedback typically contribute to establishing this spatial awareness, immersing the user in the environment. This awareness of objects and space is referred to as proprioception (Stillman, 2002). By leveraging these concepts, menu item selection can be made more intuitive, thereby enhancing the immersive quality of interactions in virtual reality. Guided by these findings, this study aimed to provide deeper insights into how these sensory inputs contribute to improved user interactions.

Therefore, this study investigated the relationship between visual cues and proprioception as a way to support menu item selection. As such, this investigation was guided by the following research question:

Main research question *What is the relationship between proprioception and reliance on visual guidance with regards to menu item selection?*

The following supporting questions were identified to help address the main research question:

Supporting question SQ1 *How can spatial awareness and familiarity with menu items be developed in a virtual environment?*

This study addressed this question by examining how spatial awareness is established in a virtual environment, drawing on existing literature as well as empirical data collected as part of this study to explore the relationship between proprioception and spatial awareness. Literature provided theoretical insights into how proprioception functions in virtual contexts, while empirical data was collected to assess how users develop and rely on spatial awareness during interaction with the virtual menu system.

Supporting question SQ2 *How can visual cues and haptic feedback support the use of proprioception to interact with menu items?*

To address this question, a menu system was designed and tested to support users in developing awareness of each menu item. Empirical data was collected to examine how the system's properties enabled users to trust their proprioceptive senses during interaction. The research documented in this paper contributes to methods of improving menu systems design for VR by specifically investigating ways to improve menu item selection. This is done by investigating the development of user behaviour as a result of familiarity with the menu system that enables users to rely on visual and proprioceptive senses to guide the selection of menu items.

2 LITERATURE REVIEW

The discussions in this section explored various areas of literature that were used to inform menu systems design and interactions in a 3D virtual environment.

2.1 Virtual reality interactions

VR technology can provide an immersive experience that has use cases in various fields of research, such as education (Lansberg et al., 2022), marine life protection and awareness (Hofman et al., 2022), psycho-therapy (Jiuwei et al., 2020), physical therapy (Lansberg et al., 2022), and medical training (Zhang et al., 2017). These use cases are typically presented in a first-person perspective, thereby simulating an embodied experience. Interacting with virtual elements in this way through VR can be perceived as natural and intuitive because these interactions mimic actions done in the real world, e.g., reaching out with a hand to touch something, which is made possible through motion-tracking in the head-mounted display (HMD) and/or two handheld controllers and handtracking technology (Kugler, 2021; Shang & Alena, 2025). This method of interaction is known as directmanipulation which enables intuitive actions that mimic physical action (Rogers et al., 2023; Shneiderman, 1983).

To further improve the ease of use for interactions, research has also been done on various methods to improve selection accuracy, which are known as disambiguation techniques. There are various techniques for disambiguation with regard to selection which can be categorised as follows (Argelaguet & Andujar, 2013):

Manual – this refers to the user selecting the virtual object that they want to with no support from the system to assist in refining the selection, i.e., the method of disambiguation relies purely on the user's own ability.

Heuristic – this method uses heuristics to assign rankings to each object and when there is more than one nearby option then an object with a higher ranking would be selected.

Behavioural – this method makes use of algorithms to dynamically assist the user to select virtual objects. This technique also has various approaches which can be further categorised into first intersected, last intersected, and closest intersected (Moore et al., 2019).

While understanding how users interact within virtual environments is crucial for shaping immersive experiences, the effectiveness of these interactions largely depends on the underlying user interface design.

2.2 User-interface (UI) elements

The graphical user-interface (GUI) improved computer interaction by enhancing text-based systems with visual elements, such as windows, icons, menus, and pointers, that enhance usability and memory (Rogers et al., 2023). A well-designed UI not only improves the user experience but also streamlines interaction with available tools. This highlights the importance of applying UI design principles to optimise and support these interactions. Various alternative interfaces have been developed to make computer interactions more intuitive, including gesture-based controls, such as swiping or dragging icons (Bragdon et al., 2011), and motion-tracking, such as eye-tracking (Menges et al., 2019). In VR, motion-tracking of body movements, such as the head and hands, enhances immersion by enabling realistic interactions (Slater, 2018).

For immersive virtual environments, these visual UI elements can be designed for greater immersion by reconsidering them as follows:

Windows – this virtual space that is typically represented as a rectangular box on a computer screen can become an entire 3D world that the user can virtually stand in and feel like that they are a part of it,

Icons – these elements, which are typically represented as small, simple images and symbols, can become 3D objects that can dynamically change size from pocket-sized for storage to life-sized objects to engage with (Ahmed et al., 2021; Park & Kim, 2022),

Menus – sets of tools provided by menus, which are typically represented as a set of panels to navigate through, can make use of metaphors like backpacks or wristwatches (Bailly et al., 2016),

Pointers – this tool used for selection is typically represented as a mouse cursor to point and click on other on-screen elements, which can evolve into intuitive tools like 3D object grabbing (Ahmed et al., 2021). The menu system is particularly notable for its role in accessing other tools.

2.3 Use of menu systems

Since the menu system is a UI element that helps users navigate a virtual environment by structuring functions (Cockburn et al., 2007), it plays an important role in supporting a user to carry out a task effectively (Chertoff et al., 2009). In the context of a 2D monitor display, the UI is limited to the monitor display. However, VR technologies immerse users in the 3D virtual environment, allowing the UI to be around the user (Calleja, 2011). Because of this, the UI is any interactable virtual object in the virtual environment, thus creating the need to reconsider how menu systems should be accessed.

Properties of a menu system, such as layout and positioning, affect the ability to learn and memorise menu items that support the user in completing a main task. Therefore, previous studies have explored menu layouts such as a linear layout which provides menu items structured as a straight list, i.e., vertical or horizontal (Chertoff et al., 2009). The linear layout follows a structure that is commonly used to present options however it comes with the limitation that some items are further away, thus making them more difficult to select (Poupyrev et al., 1996). To address this issue, a radial menu was designed that made use of a circular structure to ensure all menu items are equidistant and ensuring that all items require minimal effort to select (Komerska & Ware, 2004). Various other menu layouts have been explored in other studies such as contextual menus that display variations of items based on the context that it was activated in (Rogers et al., 2023), morphing menus that have menu items change in priority and ease of use the more often an item is used, and finger counting menus that rely on child-like finger actions for menu item layout and selection (Kulshreshth & LaViola, 2014); however, a detailed discussion of each of these is beyond the scope of this paper.

Although there are various layouts for menu systems, all of them face the same limitation, i.e., limited space without causing clutter (Shneiderman et al., 2018). For menu systems that have a lot of items, multiple layers can be used in the form of submenus where items can be grouped according to function-based categories. However, multiple layers of submenus should be approached cautiously, as it can obscure functions from users unfamiliar with their location (Bowman & Wingrave, 2001). Additionally, repetitive menu selection can lead to physical strain, which is further intensified in immersive technologies due to the large, full-body movements required, increasing fatigue (Ren & O'Neill, 2013).

For a menu system to be helpful, menu options should be accessible on-demand and easily dismissible when they are no longer needed (Bailly et al., 2016; Raskin, 2000). To adhere

to these properties, existing design solutions have attached menu systems to specific objects within the virtual environment where the options available are linked to the specific object's properties (Rogers et al., 2023). Doing so provides the user with menu options only when they want to change the properties of the specific object. However, the approach described above limits the user's access to their ability to select the object, e.g., being close enough to see or point at the object. This is particularly relevant when every object in the virtual environment has an identical set of options in its associated menu as having multiple menus will result in redundancy. To overcome this limitation and redundancy, menu systems can attach the menu to the user instead of specific objects, ensuring functions are accessible regardless of the user's location in the virtual environment. This proximity also increases user awareness of available functions.

2.4 Body-relative interactions

Body-relative interaction involves locating items relative to the body, such as finding a pen in a pocket through touch and proximity (Stillman, 2002). In virtual environments, leveraging this concept requires immersion, which fosters spatial awareness of the body and nearby objects through proprioception – a sense guided by the nervous system and motor skills, enabling intuitive and natural interactions and can improve memory and cognition (Bernard et al., 2022; Taylor, 2009). Through these sensory experiences, our spatial relationship to objects supports physical interactions and enhances cognitive functions.

Embodied cognition is a theory suggesting that cognitive processes are influenced by sensory and motor experiences, not just mental activity (Wilson & Golonka, 2013). This idea has inspired embodied design in immersive virtual environments, where physical interaction shapes how users experience and interpret the virtual world (Weijdom, 2022). How we perceive and engage with our surroundings directly impacts our thoughts and behaviours, extending to our interactions with people and objects, including those in virtual environments. This highlights the integral role of the body in shaping cognitive and experiential processes.

Proxemics, which relates to comfort based on proximity to people and objects, parallels how we experience items near our body (Hall et al., 1968). A study found that humans excel in close-range tasks, driven by proprioception and visio-tactile perception, however, immersive environments are currently optimised for interactions at arm's length (Mohanty et al., 2022), leaving interactions within closer range underexplored.

A general sense of spatial awareness allows individuals to perceive and interact with objects near their body more effectively, making these objects easier to familiarise themselves with (Mine et al., 1997). This interaction that is done without looking is referred to as eyes-off interaction (Bowman & Wingrave, 2001; Mine et al., 1997). Because of this, several studies have investigated the usage of this awareness for the virtual environment (Bernard et al., 2022; Boeck et al., 2006; Yan et al., 2018). To establish spatial awareness through proprioception, users must be able to sense the virtual objects and environment. VR technology can facilitate this awareness, which creates a sense of presence in the virtual space, allowing users to feel

immersed and more easily connect with the virtual objects (Matamala-Gomez et al., 2019; Petkova & Ehrsson, 2008).

Consideration of the user's body in designing VR interactions offers three notable benefits:

1. users have a familiar frame of reference for interactions, i.e., their own body,
2. they feel a sense of control, and
3. they can potentially perform eyes-off interaction (Hofman et al., 2022; Mine et al., 1997).

These benefits lend themselves towards supporting users in developing familiarity with a system, in this case, a menu system, which can be observed through different user behaviours.

2.5 User behaviours relating to levels of expertise

Interactions with a software system typically exhibit different user behaviours that can be associated with different levels of expertise (Shneiderman et al., 2018). Users who are learning to use a system go through different stages that can be identified through user behaviours as well (Fitts & Posner, 1967). These behaviours can be categorised as follows:

- Novice users are in the cognitive stage where they are actively learning to perform interactions which require careful attention. Because they are actively learning, their actions are intentional and slow (Ericsson & Harwell, 2019).
- Experienced users go through the associative stage, where some knowledge is established, so they focus on improving their interaction skills rather than learning to perform interactions (Cockburn et al., 2014).
- Expert users have reached the autonomous stage where their interactions are habitual due to extensive practice and their attention is no longer on the interactions with the system but rather on the task at hand (Schneider & Shiffrin, 1977; Wickens et al., 2021). Because little attention is given towards interaction with the system, this may result in multi-tasking and performing eyes-off interaction (Cockburn et al., 2014).

The user behaviours associated with expertise also apply to reliance on visual guidance. Less experienced users depend on visual cues for interactions, but as they gain confidence, this reliance decreases. Well-practiced interactions require minimal visual attention, a trait common among expert users (Schramm et al., 2016). Evidence suggests that body-relative interactions, proprioception, and visual reliance can indicate expertise progression in a system (Bowman & Wingrave, 2001; Shneiderman et al., 2018; Yan et al., 2018). For this reason, the observation of changes in user behaviours while using a menu system was considered as a method of identifying progression in expertise. Doing so provided various focal points for observations so that the user experience with a menu system could be understood. As a result,

this provided a means to understand how users progressively develop the ability to become familiar with a menu system and learn to rely on proprioception to support the interactions with the menu system. Although this has been done for interactions with virtual objects in general, to the best of the researchers' knowledge, proprioception has not yet been investigated as a means to support menu item selection in VR.

Therefore, the current study investigated proprioception and reliance on visual guidance as user behaviours to understand how these can support menu item selection in a 3D virtual environment. For the purpose of this study, user behaviours were analysed as indicators of progressing expertise in using a menu system within a virtual environment. Specifically, proprioception and reliance on visual guidance were investigated to understand how these two behaviours related to one another as expertise increased.

3 METHODOLOGY

This study adopted a qualitative approach to investigate how participants use proprioception to interact with a menu system and categorise these behaviours. More specifically, these behaviours were examined through the lens of the participants' personal experiences which were captured by observing their behaviours and self-reported feedback. Performance data were also used to verify reported experiences.

This approach, centred on the individual's experience, highlighted how users progressed from unfamiliarity with the system to expertise, with menu navigation becoming intuitive and allowing them to focus more on task completion than selecting items. Their experiences were examined through this transition, providing insights into their behaviours. To understand how participants' experiences were shaped by the VR technology used in this study, their previous experiences were cross-referenced with their current experiences as part of this study (Merriam & Tisdell, 2015).

User interactions with the Belt Menu were examined through direct, firsthand experience, allowing for a thorough exploration of how participants navigated VR menu systems. This method yielded valuable insights into interaction patterns and laid a foundation for understanding immersive interface design. Furthermore, by focusing on real-time behaviour, the study uncovered subtle proprioceptive responses, such as eyes-off interactions, that standardised questionnaires, with their inherent structure, would have been too restrictive to detect. Objective user performance data were recorded to further validate these findings and account for potential bias in self-reported performance. Discrepancies between perceived and recorded performance were then analysed to reveal underlying reasons, ensuring that participant perspectives were fully considered.

While existing studies have explored the use of proprioception in VR interactions (Boeck et al., 2006; Mine et al., 1997), there is limited empirical research specifically focused on menu system interactions. To address this gap, usability tests were conducted in a controlled environment to examine user behaviour while using a menu system to complete an assigned task.

3.1 Participants

System testing typically relies on performance metrics and larger sample sizes to identify and address specific usability issues (Barnum, 2010). These studies often prioritise generalisability, which informs the use of broader participant groups. However, appropriate sample size should be determined by the study's objective rather than a standard numerical benchmark; that is, the validity of usability studies is not solely dependent on whether a sample is large or small (Schmettow, 2012). In this study, the focus was not on pinpointing system flaws, but rather on understanding users' experiences and behaviours when interacting with a VR menu system. Accordingly, an exploratory, qualitative approach was adopted to gain insight into participants' firsthand perceptions and interactions. Qualitative research emphasises nuanced understanding over generalisability and typically employs smaller, purposefully selected samples that are sufficient once data saturation, when no new insights are observed, is reached (Guest et al., 2006).

As part of the recruitment criteria, participants were required to have prior experience with video games to ensure familiarity with virtual environments and reduce the influence of novelty on their interaction with the system. Demographic data was excluded due to ethical restrictions by the EBIT Faculty at the University of Pretoria. To collect enough reliable data on experience with a system, previous studies have shown that 3–5 participants can identify 80% of usability issues (Nielsen & Landauer, 1993), however, this approach often lacks sufficient data to determine whether these issues are frequent or isolated occurrences (Cazañas et al., 2017). Samples of 10–20 participants are thus recommended to improve the reliability of findings and achieve 90% issue detection (Faulkner, 2003; Lindgaard & Chattratchart, 2007). Accordingly, purposive sampling was employed to recruit 17 participants through personal networks who matched the aforementioned criteria. These participants represented a diverse range of experience with VR technology, from no prior exposure to professional development with head-mounted displays (HMDs) and motion-tracked controllers, ensuring a broad spectrum of skill levels within the sample.

3.2 Materials

To fulfil the study's requirements, a new menu system was developed to support body-relative interactions for menu item selection, as no existing system supported menu system interactions around the user's body. This custom system served as a research tool to investigate the relationship between proprioception and visual guidance. Participants used it to access functions and complete structured tasks, allowing for the observation and analysis of their interactions. To ensure good usability for menu systems design, one set of guidelines and one set of design goals were identified.

Drawing on concepts from various studies, four key characteristics were identified that a menu should embody to effectively sustain its intended function and behaviour (Bailly et al., 2016):

1. Menus should present a list of items that represent functions, enabling users to efficiently complete tasks (Foley et al., 1984).
2. Menu items must be presented in a visually organised structure (Dachselt & Hübner, 2007).
3. Menus should be transient, displaying information temporarily and allowing for easy dismissal (Jakobsen & Hornæk, 2007).
4. Menus should be quasimodal, requiring continuous user interaction to remain active (Raskin, 2000).

The five requirements listed below provide an appropriate goal to determine what should be expected from a new menu system (Bowman & Wingrave, 2001):

1. Users should achieve at least the same level of efficiency and accuracy as with other menu systems if not improved performance.
2. The new system should cause no significant discomfort during use.
3. The new system should not interfere with the user's ability to interact with the virtual environment.
4. The new system should be easily usable by both novice and experienced users.
5. After gaining adequate experience, users should be able to interact with the system without needing to visually focus on it, i.e., perform eyes-off interaction with the system.

To facilitate users to rely on spatial awareness that is within the immediate space around their body, menu items were designed to be in close proximity to the user so that they can take advantage of the three benefits that come from body-relative interaction (Mine et al., 1997; Yan et al., 2018):

- familiar space to easily develop spatial awareness of menu items,
- providing a sense of control and mastery,
- and being more prone to develop the ability to perform eyes-off interaction.

The resulting menu system designed for this study placed menu items around the user like a utility belt which inspired the name "Belt Menu". This interface metaphor was chosen because a utility belt is often used as a convenient way of making tools accessible for a task. To take further advantage of the utility belt metaphor, interactions within close proximity of the user would benefit from a familiar method, so virtual hands were employed as the selection

technique due to their intuitive nature for interactions such as grabbing, i.e., closing the hand, and releasing, i.e., opening the hand.

Various menu items represented functions that were provided through the Belt Menu: creating blocks of different shapes, a colouring tool, and a resizing tool. Since a belt is usually worn around the waist and it could not be assumed that all participants would be the same height, a function was also provided to adjust the height of the belt to the participant's comfort. Based on these facts, the following main menu items were provided (as shown in Figure 1):

- Measure tape Figure 1(a) – measurement is commonly used to determine the size of objects. This menu item provides tools used for changing the size of the blocks.
- Paintbrush Figure 1(b) – this represents a tool that is commonly used to apply colour to objects. This menu item provides different colours to apply to the blocks.
- Box of blocks Figure 1(c) – this represents a box from which blocks of different shapes can be retrieved.
- Arrow adjuster Figure 1(d) – this indicates the direction in which the Belt Menu can be moved. This tool is used to adjust the height of the Belt Menu.

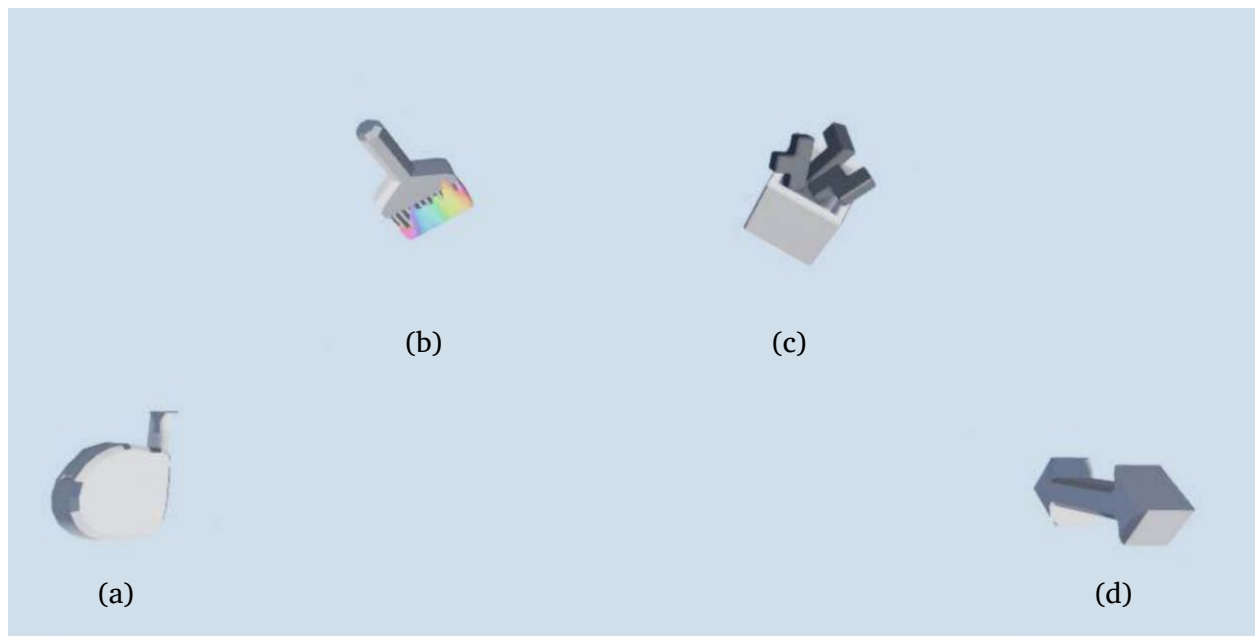


Figure 1: Belt Menu (top view)

Each menu item also provided submenu items where more specific options can be selected. These options are described in the list below. All submenu options were shaped as wheel segments since studies have shown that a radial layout for menus allows for faster selections

as all menu items are equidistant from the point of activation (Lediaeve & LaViola, 2020), i.e., the centre and the wheel segments provided large and easy targets to select. The main menu items had the following submenu options:

- Box of blocks – options to select three different shaped blocks (L | T | Z), as shown in Figure 2(a).
- Paintbrush – options to select three different coloured paintbrushes (turquoise | lime | pink), as shown in Figure 2(b).
- Measure tape – options to select either tool for enlarging (+) or shrinking (-), as shown in Figure 2(c).
- Of the four menu items, the arrow adjuster was the only main menu item that did not have submenu items as it was a tool to adjust the height of the Belt Menu.

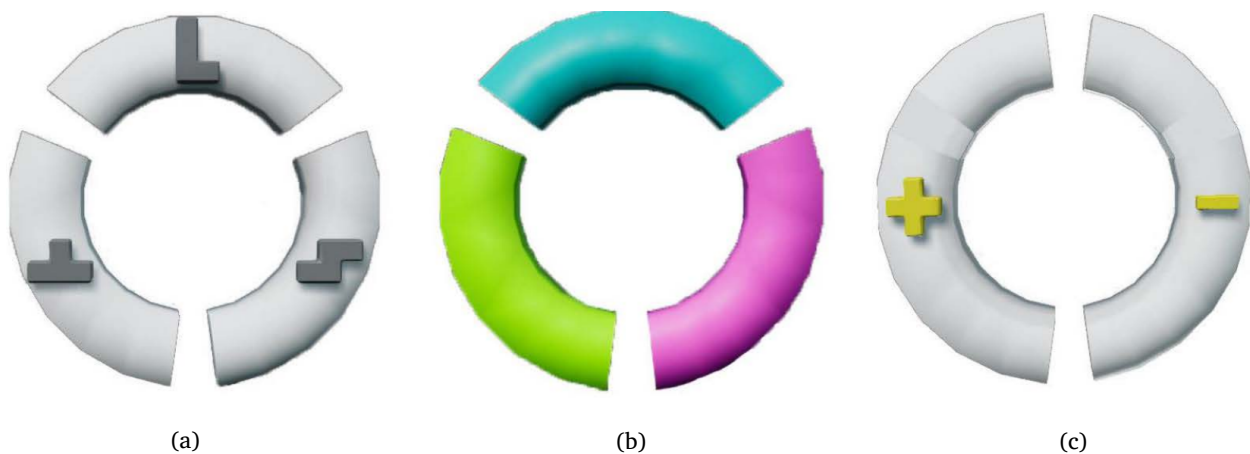


Figure 2: Submenu options

In addition to visual representation, the motion-tracked controller vibrated when touching menu items to provide haptic feedback in addition to visual feedback. The haptic feedback was provided in two ways: when touching main menu items, e.g., paintbrush, the controller would have a continuous but subtle vibration, offering a consistent tactile sensation to indicate contact; upon touching submenu items, e.g., the lime wheel segment, the controller would have a short but distinct vibration mimicking the sensation of lightly bumping a solid object to signal a change in interaction. These two different vibrations were designed to help distinguish between touching main menu items and submenu items.

To support users in developing familiarity with menu items, the Belt Menu featured consistent design elements, such as submenu items arranged like wheel segments, and incorporated haptic feedback as additional guidance. These features were provided to support users in developing habitual movements, enabling more intuitive interactions and allowing efficient navigation even without visual focus.

3.3 Procedure

To explore user behaviours with the new menu system, participants completed tasks using the Belt Menu across three separate sessions (indicated as S1, S2, and S3) that were progressively more complex. This multi-session design enabled the analysis of changes in behaviours that indicated familiarity and expertise over time, with data from each session providing insights into participants' progression. Each usability test session had a maximum duration of 30 minutes and consisted of two main sections: participants first used the Belt Menu to complete a task while their interactions were observed (10 minutes), followed by an interview to discuss their experience (20 minutes).

Observations captured real-time details of an activity (Flick, 2009), i.e., interactions with the menu system, while the think-aloud protocol encouraged participants to verbalise their thoughts during the interaction, providing a moment-by-moment account of their experiences (Pickard, 2019). Although observations usually cause participants to feel self-conscious (Rogers et al., 2023), the use of an HMD in a virtual environment concealed the researcher and possibly reduced this discomfort (Bowman & McMahan, 2007).

Interviews after each session provided deeper insights into participants' personal experiences (Flick, 2009). Participants were asked how they located menu items, their awareness of indicators such as haptic feedback, and whether the locations of menu items were easy to remember. These questions, detailed in [Appendices A and B](#), were designed to track behavioural changes and the progression of expertise. Notes were taken during each interview session.

At the start of the first session, participants were given time to interact freely with the Belt Menu to familiarise themselves with its functions. During this exploration phase, they were encouraged to select and experiment with various menu items to understand the tools available to them. This preparatory interaction ensured that participants had a foundational understanding of the Belt Menu before using it to complete the required tasks in each session.

The task to be completed required five blocks to be correctly placed in three boxes by matching the block with the label on the side of the designated boxes. The Belt Menu provided tools to generate blocks of three different shapes as well as tools to change different properties i.e., colour and size (as shown in [Figure 2](#)) of these blocks. Five blocks of each shape needed to be placed in the correct box to complete the task, resulting in a total of 15 blocks correctly placed. As each block is correctly placed in each box, the label on the side would randomly generate a new requirement of colour and shape for the next block to be matched and placed.

In session 1 (S1), the task was to match and place blocks of different shapes into boxes labelled with a specific shape (as shown in [Figure 3](#). In this session, tools for changing the colour and size were unnecessary for completing the task but these tools were still made available.

The task for session 2 (S2) was to do the same as in session 1, but additionally, the blocks would need to be in the correct shape (as shown in [Figure 4](#)).

The task for session 3 (S3) was to correctly match the shape, colour, and size of blocks that are to be placed into boxes (as shown in [Figure 5](#)).

The system utilised built-in performance tracking to record task completion times and menu

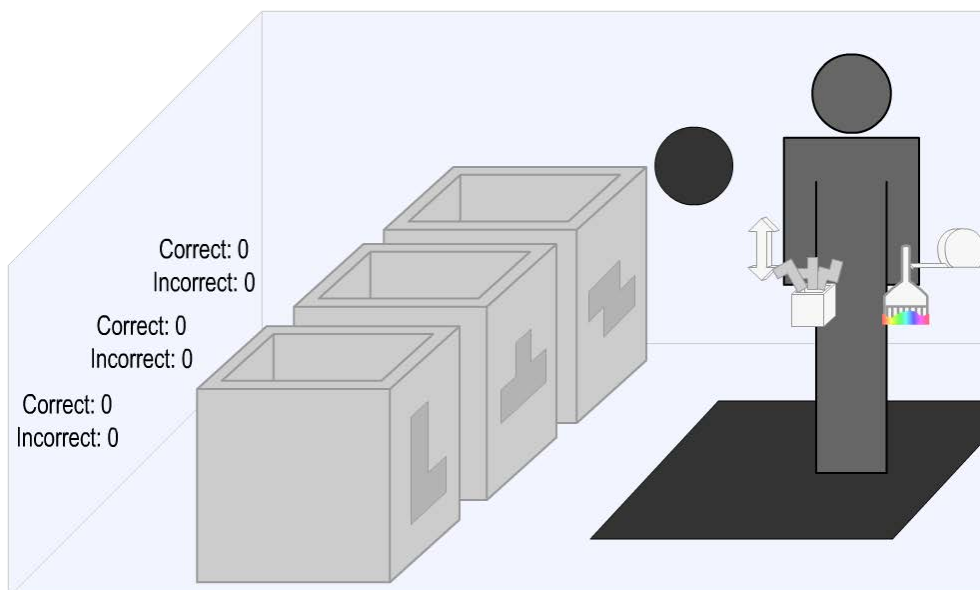


Figure 3: The virtual environment for the task in session 1 (S1)

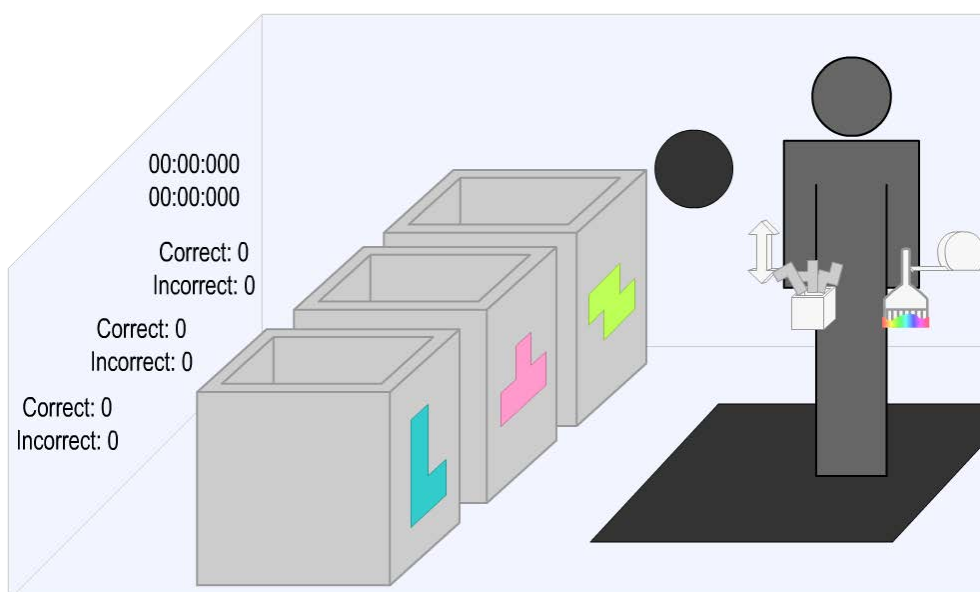


Figure 4: The virtual environment for the task in session 2 (S2)

item selections. While the number of correct and incorrect blocks placed per box was recorded, this served only to inform participants of their progress, not as a performance metric. Task timing began when participants stood on the black pad and grabbed the black sphere (Figure 5), ensuring they were positioned correctly and familiarised with grab interactions. Timing ended once five blocks of each shape were correctly placed in the corresponding boxes.



Figure 5: The virtual environment for the task in session 3 (S3)

Selection accuracy was determined by differentiating each selection as either successful or unsuccessful and was recorded as follows:

- Successful selections: Occurred when participants touched and grabbed a menu item, leading to a submenu opening or an item being selected.
- Unsuccessful selections: Occurred when a grab was attempted near a menu item but did not result in a selection.

The system identified successful and unsuccessful selections using detection areas around each main menu and submenu item (shown in Figure 6). These detection areas, represented by lines and grey areas during development, were hidden from participants during usability testing.

Selections were recorded only while the timer was active, ensuring that any actions outside the task were excluded from the menu item selection counts. Although the system utilised built-in performance tracking for selections, these records alone were considered insufficient for accurate analysis; further steps to ensure reliability are detailed in Section 4.

After completing all sessions, participants joined focus groups to reflect on their experiences and share insights. These discussions encouraged critical thinking by allowing participants to compare their experiences with those of others who interacted with the Belt Menu (Lunenburg & Irby, 2008). To ensure consistency and comparability across focus group discussions, an identical set of facilitation questions was used for each session. The complete question set is provided in Appendices A and B.

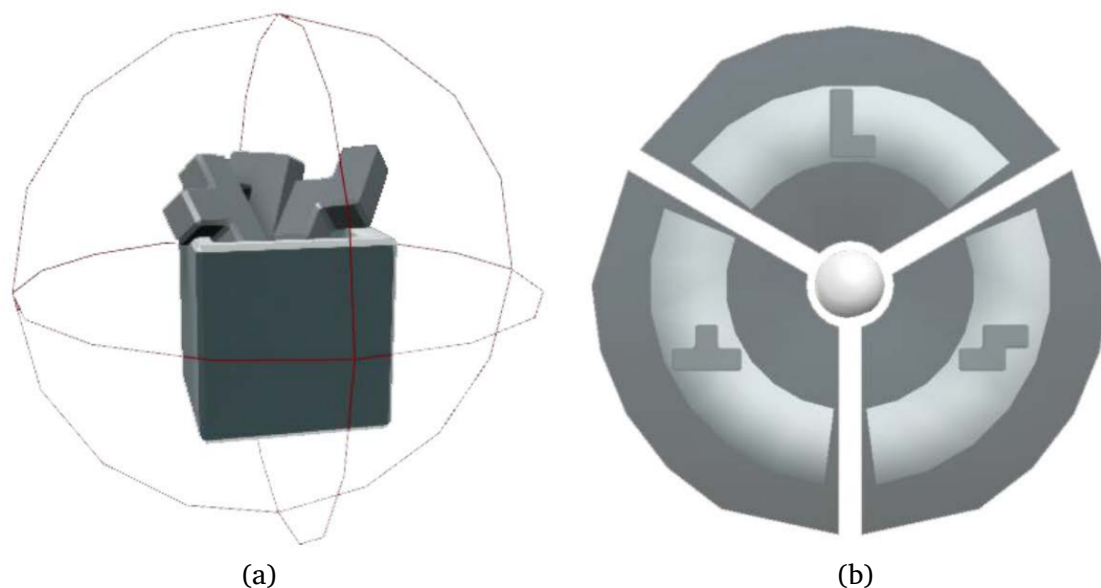


Figure 6: Nearby areas for unsuccessful selections

4 RESULTS

The results in this section draw from observations, interviews, and focus groups to analyse user behaviours and individual experiences with the Belt Menu. To complement these qualitative insights, quantitative performance data were also incorporated to validate reported experiences from a perspective beyond self-reporting.

4.1 Overall user behaviour related to expertise

The data collected from the observations, interviews, and focus groups were transcribed and imported into ATLAS.ti 22 for qualitative analysis to identify recurring themes and relationships between these themes.

Thematic analysis was used to identify user behaviours as various codes listed in Table 2. As discussed in Section 2.5, user behaviours can be related to different levels of expertise, namely: novice-, experienced-, and expert users. Therefore, these levels of expertise were used as a priori set of themes to code user behaviours across all 17 participants rather than segregating participants according to expertise. This approach ensured that the focus of this study remained on various user behaviours guided by the lens of understanding these behaviours according to expertise.

A total of 54 transcriptions (3 interviews per 17 participants plus 3 focus groups) were manually reviewed by the primary researcher to identify relevant discussions. This continuous review process allowed codes to emerge and evolve throughout the analysis to ensure accurate data representation. The process continued until all data were reviewed and no new

codes emerged. User behaviours were then colour-coded based on three themes: body-relative interactions, memory, and visual guidance, as shown in Table 1. The first two themes were identified in the literature as supporting proprioception (as discussed in Section 2.4), while visual guidance has been linked to levels of expertise.

Table 1: Colour codes of user behaviours

Theme	Code colour
Body-relative interactions	Green
Memory	Blue
Visual guidance	Purple
Hesitance	Orange

Table 2: User behaviours relating to all levels of expertise

Expertise	Codes	Occurrences
All	Double checking	39
All	Glance	86
All	Haptic	69
Novice	Clumsy	8
Novice	Direct looking	52
Novice	Forgetting	2
Novice	Function unsure	9
Novice	Looked at main menu	4
Experienced	Easy to become familiar	84
Experienced	Peripheral vision	38
Experienced	Previous session	41
Experienced	Relying on shadows	9
Experienced	Spatial awareness	8
Expert	Autopilot	9
Expert	Eyes-off interaction	45
Average		33.5

The colour coding applied to Table 2 also revealed the variety of codes identified that related to specific themes. Themes relating to body-relative interactions (orange colour coding)

had little variety, with only two user behaviours coded, but they were acknowledged to help with menu selections. User behaviours relating to memory (blue colour coding) were identified evenly across all three levels of expertise with no overlapping behaviours that all expertise experienced. However, most of the discussions occurred with experienced user behaviour, which was indicated by the number of occurrences being above the average of occurrences. There was a good variety of coded user behaviours identified (seven) that were associated with visual guidance (purple colour coding). This variety was also evenly distributed across the different levels of expertise with two codes found across all three expertise levels except for expert user behaviour where only one user-behaviour was identified.

The colour coding applied to [Table 2](#) also revealed the variety of codes identified that related to specific themes. Themes relating to body-relative interactions (orange colour coding) had little variety, with only two user behaviours coded, but they were acknowledged to help with menu selections. User behaviours relating to memory (blue colour coding) were identified evenly across all three levels of expertise with no overlapping behaviours that all expertise experienced. However, most of the discussions occurred with experienced user behaviour, which was indicated by the number of occurrences being above the average of occurrences. There was a good variety of coded user behaviours identified (seven) that were associated with visual guidance (purple colour coding). This variety was also evenly distributed across the different levels of expertise with two codes found across all three expertise levels except for expert user behaviour where only one user-behaviour was identified.

All participants were generally expected to exhibit novice behaviour in session 1 since this was their first time using the Belt Menu. In session 2, user behaviours were investigated to identify any resemblance of associative behaviour, i.e., having some established knowledge and exploring new ways to improve their interactions. In session 3, user behaviours were investigated to identify any resemblance of autonomy that is prevalent among experts, i.e., instinctive interactions with little attention required. Some behaviours were prevalent during all three sessions even though progression in expertise could be observed. The behaviours that were prevalent in all sessions are listed in [Table 2](#) and labelled as “All”. The themes of various user behaviours identified and listed in [Table 1](#) will be discussed below. The order of these discussions begins with more general observations regarding expertise and narrows down the focus towards themes that inform the use of proprioception and visual guidance thereby, linking back to addressing the research questions.

4.2 User performance

User performance data were collected to complement observations and self-reporting by offering additional insights into user behaviour. The focus was on measuring individual progression in expertise rather than comparing participants. This section highlights data from sessions 1 and 3, showcasing participants’ initial and final interactions with the Belt Menu.

Two performance metrics were analysed: task completion time and selection accuracy. Task completion time was recorded in every session to evaluate efficiency. As tasks became

progressively more complex across sessions, longer completion times were anticipated. The median times for each session were as follows:

- Session 1 – 00:33.362
- Session 2 – 00:54.031
- Session 3 – 01:37.744

The median was used to minimise the influence of outliers and ensure a fair comparison across the three sessions. As each session introduced additional tools, task complexity increased, leading to expected completion time increases, roughly doubling in session 2 and tripling in session 3. Selection accuracy was determined by successful or unsuccessful selections, as discussed in [Section 3.3](#). However, this metric does not account for successful but incorrect selections, such as selecting the wrong size or colour, e.g., choosing a small pink L-shaped block instead of the required large lime one.

Therefore, to accurately assess selection data, an ideal benchmark for required selections was established. In session 1, participants needed to place five blocks of each shape (L, T, Z) into the correct boxes, totalling 15 blocks. Generating each block required two selections: one from the main menu (box blocks) and one from the submenu (wheel segment for a shape), resulting in the minimum required selections (MRS) being 30. Any selections beyond this would be considered unnecessary, leading to inefficiency. Unnecessary selections could arise from selecting the wrong menu item or changing a block's colour unnecessarily. These additional selections, while counted as successful, were included in the total number of selections (TNS). This TNS is also the total for all 17 participants for each of the three sessions, and therefore, the MRS should then also be multiplied by 17 sessions. The following formula was used to calculate the percentage of unnecessary selections:

$$\% \text{ of unnecessary selections} = \frac{\text{TNS} - (\text{MRS} \times 17 \text{ sessions})}{\text{TNS}} \quad (1)$$

This formula calculates inaccuracy to measure how selections contributed expertise progression over the three sessions. [Figure 7](#) shows the percentage of unnecessary selections, and the trend, for sessions 1 to 3.

For session 2, the task required an added selection for the paintbrush tool, doubling the required selections, resulting in 60 (15×4). The task for session 3, which required both colour and size adjustments, tripled the selections to 90 (15×6). Despite the task in session 3 being more than three times as complex as in session 1, the percentage of unnecessary selections decreased from 56.51% to 51.65% (a 4.86% reduction), indicating improved efficiency. This suggests participants became more familiar with the Belt Menu, likely developing expert user behaviours that enhanced their performance.

The user performance data discussed above regarding completion time and accuracy both indicated positive evidence that participants felt more familiar with the Belt Menu and were able to increase efficiency as a result.

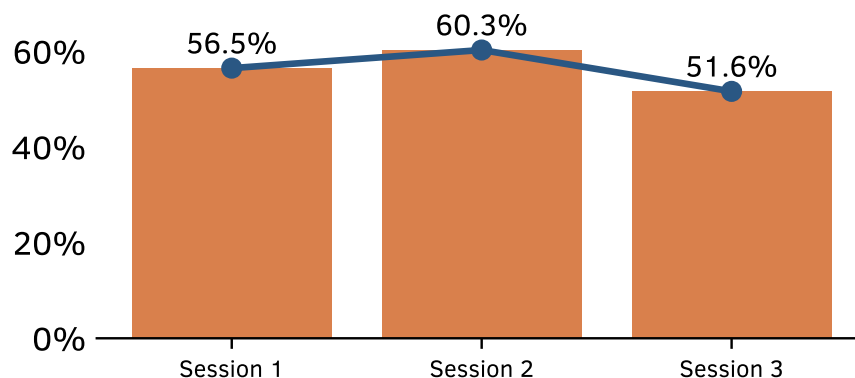


Figure 7: Unnecessary selections for all three sessions

4.3 Hesitance

Hesitance, as exhibited by participants, refers to uncertainty about what action to take or how to perform it. For example, a participant might look at a menu item resembling a measuring tape but pause to contemplate the function it represents. The following discussion examines the causes of this user behaviour as observed during interactions with the Belt Menu.

In session 1, participants often felt clumsy with their interactions if they were not intentionally focusing on their selections, specifically with main menu items. During the interviews, Participants P02 and P10 discussed that they were unsure of some menu items' functions at first, especially the measuring tape, but were able to understand the function once they started using them. Participant P02 said: *"Measuring tape (ruler) is more to measure length than size."* and that they *"... would've preferred a block with arrows going up and down. There could be a more elegant solution. [It] can be better thought out."*

Participants P14 and P17 stated that they understood the function of the paintbrush but were unsure how to use this tool. During their session 1, participant P14 tried to use the paintbrush on multiple virtual objects, including the boxes and other menu items, before using them on the blocks. All 17 participants also mentioned that they struggled to remember the specific positions of the submenu for the different shapes so they had to look at the menu items before selecting the blocks from the Belt Menu.

This behaviour indicated unfamiliarity and served as a baseline for recording user behaviour. While it does not directly address the research questions, it establishes a reference point for tracking potential changes in familiarity over future sessions.

4.4 Haptic feedback

Haptic feedback, in this context, refers to the vibrations generated by the controller during interactions with the Belt Menu. The discussion below explores the user behaviours influenced by this feedback, as well as its impact on their impressions of the Belt Menu and its role in assisting them with task completion across sessions.

Even though haptic feedback was designed to provide feedback in addition to visual feedback, only some participants mentioned that they noticed the controllers “buzzing”, i.e., providing haptic feedback, when touching menu items. Participant P14 in particular found this as a helpful confirmation to indicate that they touched the menu items by stating that the haptic feedback “*gives a good feeling that you [have] really touched it. It gives an idea that you already reached there. Not a very hard vibration. [The vibration] makes it more real.*”. Others (P05, P11, P16, P17) stated that the “buzzing” was annoying and sometimes made them feel overwhelmed. In post-session focus groups, participants noted that haptic feedback might carry a negative connotation, as some video games use it to signal errors, such as crashing or taking damage. Participant P11 was aware of this connotation before being prompted and made a mental shift between sessions 1 and 2, leading them to use the haptic feedback to help them feel the menu items. Some other participants (P01, P06, P07, P08, P10) confirmed during the interviews and focus group discussions that they were only confident of their actions without looking because the controller provided haptic feedback when touching menu items.

In contrast, participant P03 stated “*I couldn’t recall after a session whether the controllers vibrated or not or anything highlighted. I just waved in the direction of whatever I wanted*” which suggested that they did not rely on non-visual cues but instead relied only on well-practised movement and memory.

To summarise, haptic feedback was incorporated as a complementary cue for visual representation. Discussions with participants provide a variety of evidence regarding the use of haptic feedback to support menu item selection. This design directly aligns with research sub-question SQ2, which partly examines how haptic feedback influences user behaviours. The findings regarding haptics revealed three distinct user sentiments: helpful, annoying, and unnoticed.

4.5 Body-relative interactions and memory

This section examines user behaviours resulting from having all menu items positioned within close proximity, i.e., within arm’s reach, and explores how this arrangement contributed to their ability to recall the location of each function.

In session 1, all participants found the task straightforward, enabling them to repeatedly practice interactions with the Belt Menu. In session 2, they easily adapted to using the paintbrush, as its layout and selection mirrored prior interactions. Participant feedback, such as “*Yes, the previous session helped a lot for feeling familiar with the menu system today*” (Participant P10), indicated that familiarity and memory from prior sessions contributed to their ease of use. Participant P16 similarly expressed “*Today’s session I did not have to figure things out and just go ahead and use it. So that experience did help*”.

During interviews, participants consistently stated that exposure in previous sessions made the Belt Menu more intuitive, helping them better recall submenu item positions in session 2. This progression reflects the movement from novice to experienced users, aligning with the associative stage of expertise. Session 3 required the use of all menu items, demanding

participants to resize and recolour blocks before placement, which involved frequent switching between tools. Despite this complexity, participants focused on task completion rather than learning new interactions. Many participants (P01, P04, P08, P10) reported feeling confident and efficient, with selections described as feeling like “*autopilot*.” Participant P01 stated, “*I naturally just waved my hand into the option that I wanted*,” and Participant P17 noted, “*I instinctively moved my hand into the right position*.” Several participants (P01, P03, P06, P07, P08, P10) reported performing eyes-off interactions, i.e., selecting items without looking, especially in general directions or when using submenu items. These behaviours reflect proficiency, as users required minimal attention for menu interactions, achieving a level of autonomy.

The evidence presented in this section on body-relative interactions and memory contributes to understanding how familiarity with nearby virtual objects can be established within a virtual environment, as outlined in research sub-question SQ1.

4.6 Reliance on visual guidance

This section discusses various user behaviours related to the use of visual cues for navigating the Belt Menu.

In session 1, all participants took time to familiarise themselves with the Belt Menu, carefully observing each menu item’s feedback, such as blocks changing colour when touched with a paintbrush, and interacting with other objects. This behaviour aligns with the novice cognitive stage, where users intentionally learn system interactions.

In session 2, new behaviours emerged where participants P04, P05, P07, P10, P15, P16, and P17 frequently glanced at their selections to confirm choices, often noticing submenu items from “*the corner of [their] eye*”, i.e., their peripheral vision. Participants P02, P14, and P15, who preferred to inspect every interaction, reported that as tasks became more complex, they relied even more on visual confirmation. They noted this behaviour also extended to their everyday routines.

During the focus groups, participants P04, P05, P07, P15, and P17 discussed how selecting submenu items felt easier with less visual guidance compared to main menu items. Observational data corroborates this by showing that participants typically glanced at the main menu items before focusing on finalising selections in the submenu. This behaviour indicates that, with experience, participants relied less on visual cues, often without conscious awareness. However, the exact reasons for this shift in reliance could not be conclusively determined from the data.

Previous studies have shown that as users gain expertise, they rely less on visual guidance, with expert users performing eyes-off interactions (Cockburn et al., 2014; Ericsson & Harwell, 2019; Schramm et al., 2016). Consistent with these findings, participants in earlier sessions carefully monitored each interaction with the Belt Menu, often looking directly at menu items. As familiarity grew, participants became more confident, initially glancing, and then using peripheral vision to confirm selections. Eventually, participants reached a stage where they could select items without relying on sight, demonstrating eyes-off interaction. Observing

these behaviours resulted in identifying four categories of user behaviours regarding visual guidance: directly looking, glancing, peripheral vision, and eyes-off interaction (as shown in Table 2 in Section 4.1). Overall, the results showed that increased familiarity allowed participants to use a broader range of visual guidance strategies. Figure 8 illustrates how these categories correspond to varying levels of familiarity.



Figure 8: A representation of the range of visual guidance in relation to familiarity

The findings on visual guidance in this section contribute to understanding user behaviours that reflect the development of familiarity with virtual objects in a virtual environment, addressing research sub-questions SQ1 and SQ2. Additionally, they provide evidence of how visual cues support the use of proprioception in menu item selection, further addressing sub-question SQ2.

4.7 Use of shadows

The following discussion outlines how user behaviour emerged from participants noticing shadows in the virtual environment, which enhanced their awareness of menu items to be selected.

Shadows in the virtual environment were intended solely for realism, yet three participants used them to locate and interact with menu items. This suggests that shadows could function beyond aesthetics, similar to physical environments that indicate object presence and movement.

In session 2, participants P08 and P10 noticed shadows cast by the menu items and their virtual hands, which helped guide their selections when they were not looking directly at the menu. Participant P10 noted, “*Having the shadows really helped understand where everything is without looking directly at the menu items.*” Participant P05 also observed the shadows, which helped them identify two menu items (measure tape, arrow adjuster) that were out of sight when looking forward.

The discussion on how shadows aid in menu item awareness highlights that indirect perceptual properties can support interactions within a virtual menu system, addressing research sub-question SQ2.

4.8 Concluding user behaviours

The results stated above provide evidence that participants were unfamiliar with the Belt Menu at first but as they progressed over three sessions, they noticed more aspects of the Belt Menu

and became more familiar over time. This can be expected, however, the changes in the user behaviours linked to the progression of familiarity are of particular interest.

The Belt Menu's close proximity to the user significantly enhanced accessibility and memory, enabling participants to select menu items intuitively. While haptic feedback was intended to support awareness, its effectiveness varied among users. Notably, visual cues were central to interaction, with behaviours ranging from direct looking to eyes-off interactions evolving as familiarity increased. Additionally, some participants leveraged shadows as an unanticipated cue to locate menu items. These findings suggest that integrating multiple feedback mechanisms, particularly those that enhance visual guidance, can foster familiarity and support more efficient, intuitive interaction with VR menu systems.

5 DISCUSSION

The results above highlighted user behaviours that were identified and linked to progression in expertise. These user behaviours were analysed further to address the research questions posed in [Section 1](#). The first two subsections below provide discussions that address supporting research questions. Based on these two subsections, the main research question will be addressed.

5.1 Addressing supporting question SQ1

How can spatial awareness and familiarity with menu items be developed in a virtual environment?

Because proprioception is a sense of spatial awareness and familiarity with the space and objects around an individual, it is important to discuss how this sense can be supported. The Belt Menu was intentionally designed to have menu items around the user's body, which is a space that the user will already find familiar and virtual objects in this space will always be accessible.

This study suggests that, while participants initially showed hesitation in the first session, they developed familiarity with menu item positions around their body through practice, improving memory and recall in subsequent sessions. Using the lens of user behaviours associated with the levels of expertise as discussed in [Section 2.5](#); the results of this study yielded similar evidence with the Belt Menu as well, inferring that familiarity can be developed with a menu system within close proximity in a virtual environment. Through practice, participants developed instinctive selection movements, a key indicator of expert behaviour. This progression of familiarity was further evident as some participants reduced their reliance on visual guidance as they grew more familiar with the menu system. In addition to the evidence extracted from user behaviour, the performance data for time completion and selection accuracy provides evidence of familiarity as these showed increased efficiency over sessions.

The instinctive selection of menu items, requiring minimal conscious effort, indicates that participants have developed a robust sense of spatial awareness around their body. Learning

and memorising menu items were further supported by haptic feedback, as discussed in the next supporting question.

5.2 Addressing supporting question SQ2

How can visual cues and haptic feedback support the use of proprioception to interact with menu items?

When interacting with a virtual environment, one of the most salient forms of feedback for interaction with virtual objects is visual feedback, e.g., selecting a menu item results in a tool that is visibly selected and placed in the user's hand. Visual feedback enables users to associate a function with its spatial position, thereby enhancing memory retention. This effect is particularly evident among novice users, who rely more heavily on visual cues during the earlier sessions. An unexpected user behaviour relating to the use of visual feedback was identified, which was the use of shadows. Because shadows behave in the same way in a virtual environment, some participants realised that they could rely on the virtual shadows to guide their hands toward menu items for selections when menu items were not in sight. Therefore, shadows can be used as visual information for users to be aware of their virtual surroundings as well. Although this visual cue was used in an unintended way, this still provides evidence that visual cues support the use of proprioception for interacting with a menu system.

A less prominent form of feedback that was mostly used to supplement other forms of feedback is haptics. While haptic feedback was used in the Belt Menu to create awareness of menu items when they are touched, evidence shows that there were four sentiments: some found it to be informative, annoying, unnoticed, or preferred to rely on practised movement and memory instead. Based on these results, haptic feedback is not essential for menu system design, but it can enhance users' awareness of nearby menu items during hand movements, complementing visual guidance.

The results also suggest that there is a range of user behaviours that were identified with regard to the reliance on visual feedback to guide menu item selection. These findings are further discussed in the next subsection to address the primary research question.

5.3 Addressing the main research question

What is the relationship between proprioception and reliance on visual guidance with regard to menu item selection?

One of the key findings discussed in the results is that as familiarity with the menu system increases, users rely less on visual cues when selecting menu items. Two additional user behaviours were observed to extend this understanding further. First, some participants felt more confident selecting main menu items without looking but were less confident with submenu items, likely because selecting a submenu finalises the action, requiring more care. Another possibility is that submenu item selections are less practised since they require first selecting a main menu item. Second, some participants always looked at selections to reduce error,

indicating a personal preference. This could relate to the “paradox of the active user” (Carroll & Rosson, 1987), where users resist adopting more efficient methods due to the risk of temporary performance dips.

Overall, user behaviours regarding visual guidance appear more tied to personal preference than expertise progression. Participants in later sessions had more experience, helping them to rely less on visual guidance, resulting in them using visual cues in a broader range of ways. For example, an experienced user might focus directly on a primary menu item while quickly glancing at submenu options. They also rely on peripheral vision to stay aware of additional choices and can even select another menu item with their other hand without looking, demonstrating all four categories of visual guidance.

In contrast, less experienced users are more restricted in using visual guidance, relying on it more consistently. However, this trend is not universal because some experienced users still prefer to look at selections to ensure success.

These results thus suggest that placing menu items close to the body enhances memorability and accessibility, making selection more intuitive in virtual environments. Familiarity with the interface develops through visual and haptic feedback, with haptic feedback as a supplementary role. Visual guidance is key in developing proprioception, allowing users to locate menu items near their body. Users’ reliance on visual feedback decreases as they become more familiar with the menu, ultimately enabling more efficient, eyes-off interaction.

Familiarity aids proprioception in menu selection, with visual guidance helping memorise the locations of virtual objects. Over time, as spatial awareness improves, users rely less on visual cues, further supporting proprioception in selecting menu items.

5.4 Limitations

By taking a qualitative approach, this study focused on insight extracted from the experience of individuals, thus making the outcomes of this study non-generalisable by nature. Making use of this approach also inherently made use of a small sample size as the focus of the study was based on understanding the richness of the experiences.

The researchers were not permitted to collect basic demographics such as age or gender data due to institutional ethics policies; therefore, any inferences that could be based on such data were not possible. Additionally, since recruitment criteria included experience with video games, the study did not explore the learnability of the Belt Menu for users with lower technological proficiency than the included participants.

Bimanual interaction, i.e., the use of two hands for interactions, was acknowledged as part of this study since the Belt Menu required the use of both hands. Although hand dominance was investigated as part of a larger study (Ka et al., 2023), it is not extensively discussed here since it was not the focus of this study.

The validity of the data analysis was supported through triangulation by incorporating multiple forms of data collection. However, the study’s outcomes could have been further improved by employing additional reliability measures, such as inter-rater reliability.

5.5 Implications and future work

This study highlights a direct link between visual guidance and proprioception in supporting menu item selection within immersive virtual environments. The findings of this study suggest that developing spatial awareness through well-practised movements is more important than haptic feedback for fostering proprioception. Findings showed that reduced reliance on visual guidance complemented proprioception in selecting menu items. These insights provide a foundation for future studies to use visual guidance categories to better understand user interactions with menu systems in 3D environments. Future studies can build on these findings by conducting research that quantitatively measures user performance, establishing a benchmark for comparison with other menu systems in immersive virtual environments.

The Belt Menu was designed for forward-facing tasks, but future studies could explore body-tracking menu systems for multi-directional movement, such as rotating utility belts in VR games, to enhance menu selection through proprioception. Since multi-directional tasks are likely more complex, future studies can use this condition to test the Belt Menu's feasibility in supporting users with intricate tasks, providing empirical insights into its effectiveness.

Some participants noted that menu items obstructed their view, particularly when the Belt Menu was positioned too high. While adjusting the height resolves this, a more innovative solution could involve dynamically adjusting the transparency of menu items based on the proximity of the user's hands. Exploring such designs could yield valuable insights into the optimal distance from the user's body, which could help determine when this approach becomes necessary to prevent visual obstruction.

An unexpected finding from this study was participants leveraging shadows to enhance selection and awareness in an immersive virtual environment. This suggests that shadows may serve purposes beyond increasing realism, potentially improving interaction design. Since this study focused on selection within arm's reach, future research could explore how shadows guide selection techniques beyond this range, such as the Go-Go Interaction Technique (Poupyrev et al., 1996), where shadows remain visible even when the virtual hand is obscured.

6 CONCLUSION

This study set out to investigate the relationship between proprioception and visual guidance and how these two senses can be used to support menu item selection in a 3D virtual environment. To do so, a menu system was created using concepts identified from existing literature, which included body-relative interactions, user behaviours relating to expertise, and levels of expertise relating to the use of visual guidance. This menu system was then used by participants in several usability test sessions so that their user experiences and behaviours could be investigated for progress over time.

The results provided evidence that participants primarily relied on experience and well-practised movements to establish spatial awareness and familiarity with menu items, which allowed them to rely less on visual guidance. Furthermore, with regards to reliance on visual

guidance, the findings built on existing literature by identifying a range of four categories: directly looking, glancing, use of peripheral vision, and eyes-off interaction. This range was inversely related to familiarity, with a strong reliance on visual guidance when users were unfamiliar with the system. As familiarity increased, users transitioned toward proprioceptive interaction, enabling more intuitive menu selection from a body-centred interface.

7 ACKNOWLEDGMENTS

This research utilised ChatGPT versions 3.5 and 4 to enhance the clarity and phrasing of the written work. This was done by the author writing the discussion in their own words and then using ChatGPT to rephrase the written work. However, all content remains the original conceptual work of the authors, and ChatGPT was not used to generate any original text.

References

- Ahmed, N., Lataifeh, M., & Junejo, I. (2021). Visual pseudo haptics for a dynamic squeeze/grab gesture in immersive virtual reality. *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*, 1–4. <https://doi.org/10.1016/j.cag.2012.12.00310.1109/ICHMS53169.2021.9582654>
- Argelaguet, F., & Andujar, C. (2013). A survey of 3D object selection techniques for virtual environments. *Computers & Graphics*, 37(3), 121–136. <https://doi.org/10.1016/j.cag.2012.12.003>
- Bailly, G., Lecolinet, E., & Nigay, L. (2016). Visual menu techniques. *ACM Computing Surveys*, 49(4), 1–41. <https://dl.acm.org/doi/10.1145/3002171>
- Barnum, C. M. (2010). *Usability testing essentials: Ready, set ... test!* Elsevier. <https://www.sciencedirect.com/book/9780123750921/usability-testing-essentials>
- Bernard, C., Monnoyer, J., Ystad, S., & Wiertlewski, M. (2022). Eyes-off your fingers: Gradual surface haptic feedback improves eyes-free touchscreen interaction. *CHI Conference on Human Factors in Computing Systems*, 1–10. <https://doi.org/10.1145/3491102.3501872>
- Boeck, J. D., Raymaekers, C., & Coninx, K. (2006). Exploiting proprioception to improve haptic interaction in a virtual environment. *Presence: Teleoperators and Virtual Environments*, 15(6), 627–636. <https://doi.org/10.1162/pres.15.6.627>
- Bowman, D. A., & McMahan, R. P. (2007). Virtual reality: How much immersion is enough? *IEEE Xplore, Computer*, 40(7), 36–43. <https://doi.org/10.1109/MC.2007.257>
- Bowman, D. A., & Wingrave, C. A. (2001). Design and evaluation of menu systems for immersive virtual environments. *Proceedings IEEE Virtual Reality 2001*, 149–156. <https://doi.org/10.1109/VR.2001.913781>

- Bragdon, A., Nelson, E., Li, Y., & Hinckley, K. (2011). Experimental analysis of touch-screen gesture designs in mobile environments. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 403–412. <https://doi.org/10.1145/1978942.1979000>
- Calleja, G. (2011). *In-game: From immersion to incorporation*. MIT Press. <https://direct.mit.edu/books/monograph/2889/In-GameFrom-Immersion-to-Incorporation>
- Carroll, J. M., & Rosson, M. B. (1987). Paradox of the active user. In J. M. Carroll (Ed.), *Interfacing thought: Cognitive aspects of human-computer interaction* (pp. 80–111). MIT Press. <http://dl.acm.org/citation.cfm?id=28446.28451>
- Cazañas, A., de San Miguel, A., Parra, E., Cazañas, A., de San Miguel, A., & Parra, E. (2017). Estimating sample size for usability testing. *Enfoque UTE*, 8, 172–185. <https://doi.org/10.29019/enfoqueute.v8n1.126>
- Chertoff, D. B., Byers, R. W., & LaViola, J. J. (2009). An exploration of menu techniques using a 3D game input device. *Proceedings of the 4th International Conference on Foundations of Digital Games*, 256–262. <https://doi.org/10.1145/1536513.1536559>
- Cockburn, A., Gutwin, C., & Greenberg, S. (2007). A predictive model of menu performance. *Conference on Human Factors in Computing Systems – Proceedings*, 627–636. <https://doi.org/10.1145/1240624.1240723>
- Cockburn, A., Gutwin, C., Scarr, J., & Malacria, S. (2014). Supporting novice to expert transitions in user interfaces. *ACM Computing Surveys*, 47(2), 31:1–31:36. <https://doi.org/10.1145/2659796>
- Dachselt, R., & Hübner, A. (2007). Three-dimensional menus: A survey and taxonomy. *Computers & Graphics*, 31(1), 53–65. <https://doi.org/10.1016/J.CAG.2006.09.006>
- Ericsson, K. A., & Harwell, K. W. (2019). Deliberate practice and proposed limits on the effects of practice on the acquisition of expert performance: Why the original definition matters and recommendations for future research. *Frontiers in Psychology*, 10, 1–19. <https://doi.org/10.3389/fpsyg.2019.02396>
- Eriksson, M. (2016). *Reaching out to grasp in virtual reality: A qualitative usability evaluation of interaction techniques for selection and manipulation in a VR game* [Masters thesis]. KTH Royal Institute of Technology. Retrieved October 31, 2017, from <http://www.diva-portal.org/smash/record.jsf?pid=diva2:946219>
- Faulkner, L. (2003). Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, 35(3), 379–383. <https://doi.org/10.3758/BF03195514>
- Fitts, P. M., & Posner, M. I. (1967). *Human performance* (2nd ed.). Brooks/Cole Publishing Company. https://books.google.co.za/books/about/Human_Performance.html?id=XtFOAAAAMAAJ
- Flick, U. (2009). *An introduction to qualitative research* (4th ed.). SAGE. https://books.google.co.za/books/about/An_Introduction_to_Qualitative_Research.html?id=bn73UEA3thIC

- Foley, J. D., Wallace, V. L., & Chan, P. (1984). Human factors of computer graphics interaction techniques. *IEEE Computer Graphics and Applications*, 4(11), 13–48. <https://doi.org/10.1109/MCG.1984.6429355>
- Guest, G., Bunce, A., & Johnson, L. (2006). How many interviews are enough?: An experiment with data saturation and variability. *Field Methods*, 18(1), 59–82. <https://doi.org/10.1177/1525822X05279903>
- Hall, E. T., Birdwhistell, R. L., Bock, B., Bohannon, P., Diebold, Durbin, M., Edmonson, M. S., Fischer, J. L., Hymes, D., Kimball, S. T., La Barre, W., McClellan, J. E., Marshall, D. S., Milner, G. B., Sarles, H. B., Trager, G. L., & Vayda, A. P. (1968). Proxemics [and comments and replies]. *Current Anthropology*, 9(2), 83–108. <https://doi.org/10.1086/200975>
- Hofman, K., Walters, G., & Hughes, K. (2022). The effectiveness of virtual vs real-life marine tourism experiences in encouraging conservation behaviour. *Journal of Sustainable Tourism*, 30(4), 742–766. <https://doi.org/10.1080/09669582.2021.1884690>
- Jakobsen, M. R., & Hornæk, K. (2007). Transient visualizations. *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, 69–76. <https://doi.org/10.1145/1324892.1324905>
- Jiuwei, L., Haoyu, Y., Fei, L., & Jiede, W. (2020). Application of virtual reality technology in psychotherapy. *2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, 359–362. <https://doi.org/10.1109/ICHCI51889.2020.00082>
- Ka, K. S., Bosman, I., & Bothma, T. (2023). *Using proprioception to guide menu item selection in virtual reality* [Masters dissertation]. University of Pretoria. <https://repository.up.ac.za/items/da0a2d1e-4f3b-4c14-a3ce-b8a87ad75982>
- Komerska, R., & Ware, C. (2004). A study of haptic linear and pie menus in a 3D fish tank VR environment. *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 224–231. <https://doi.org/10.1109/HAPTIC.2004.1287200>
- Kugler, L. (2021). The state of virtual reality hardware. *Communications of the ACM*, 64(2), 15–16. <https://doi.org/10.1145/3441290>
- Kulshreshth, A., & LaViola, J. J. (2014). Exploring the usefulness of finger-based 3D gesture menu selection. *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, 1093–1102. <https://doi.org/10.1145/2556288.2557122>
- Lansberg, M. G., Legault, C., MacLellan, A., Parikh, A., Muccini, J., Mlynash, M., Kemp, S., Buckwalter, M. S., & Flavin, K. (2022). Home-based virtual reality therapy for hand recovery after stroke. *PM&R*, 14(3), 320–328. <https://doi.org/10.1002/pmrj.12598>
- Lediaeva, I., & LaViola, J. (2020). Evaluation of body-referenced graphical menus in virtual environments. *Proceedings of Graphics Interface 2020*, 308–316. <https://doi.org/10.20380/GI2020.31>
- Liimatainen, K., Latonen, L., Valkonen, M., Kartasalo, K., & Ruusuvuori, P. (2021). Virtual reality for 3D histology: Multi-scale visualization of organs with interactive feature

- exploration. *BMC Cancer*, 21(1), 1133. <https://doi.org/10.1186/s12885-021-08542-9>
- Lindgaard, G., & Chattrachart, J. (2007). Usability testing: What have we overlooked? *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1415–1424. <https://doi.org/10.1145/1240624.1240839>
- Lunenburg, F. C., & Irby, B. J. (2008). *Writing a successful thesis or dissertation: Tips and strategies for students in the social and behavioral sciences*. Corwin Press. <https://us.sagepub.com/en-us/nam/book/writing-successful-thesis-or-dissertation>
- Matamala-Gomez, M., Donegan, T., Bottiroli, S., Sandrini, G., Sanchez-Vives, M. V., & Tassorelli, C. (2019). Immersive virtual reality and virtual embodiment for pain relief. *Frontiers in Human Neuroscience*, 13, 1–12. <https://doi.org/10.3389/fnhum.2019.00279>
- Menges, R., Kumar, C., & Staab, S. (2019). Improving user experience of eye tracking-based interaction: Introspecting and adapting interfaces. *ACM Transactions on Computer-Human Interaction*, 26(6), 1–46. <https://doi.org/10.1145/3338844>
- Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation* (4th ed.). John Wiley & Sons. https://books.google.co.za/books/about/Qualitative_Research.html?id=JFN_BwAAQBAJ
- Mine, M. R., Brooks, F. P., & Sequin, C. H. (1997). Moving objects in space: Exploiting proprioception in virtual-environment interaction. *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, 19–26. <https://doi.org/10.1145/258734.258747>
- Mohanty, R. R., Raina, A. S., Chaudhuri, S., Quek, F., Sueda, S., & Krishnamurthy, V. R. (2022). Spatial manipulation in virtual peripersonal space: A study of motor strategies. *Journal of Computing and Information Science in Engineering*, 23(21004). <https://doi.org/10.1115/1.4054277>
- Moore, A., Kodeih, M., Singhanian, A., Wu, A., Bashir, T., & McMahan, R. (2019). The importance of intersection disambiguation for virtual hand techniques. *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 310–317. <https://doi.org/10.1109/ISMAR.2019.00029>
- Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, 206–213. <https://doi.org/10.1145/169059.169166>
- Park, S., & Kim, Y. (2022). A metaverse: Taxonomy, components, applications, and open challenges. *IEEE Access*, 10, 4209–4251. <https://doi.org/10.1109/ACCESS.2021.3140175>
- Petkova, V. I., & Ehrsson, H. H. (2008). If I were you: Perceptual illusion of body swapping. *PLOS ONE*, 3(12), 1–9. <https://doi.org/10.1371/journal.pone.0003832>
- Pickard, A. J. (2019). *Research methods in information* (2nd ed.). Facet Publishing. <https://www.cambridge.org/core/books/research-methods-in-information/D922B9D95E51D50CDC0A7DB5C3765336>

- Poupyrev, I., Billinghamurst, M., Weghorst, S., & Ichikawa, T. (1996). The go-go interaction technique: Non-linear mapping for direct manipulation in VR. *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, 79–80. <https://doi.org/10.1145/237091.237102>
- Raskin, J. (2000). *The humane interface: New directions for designing interactive systems*. Addison-Wesley Professional. <https://www.amazon.com/Humane-Interface-Directions-Designing-Interactive/dp/0201379376>
- Ren, G., & O'Neill, E. (2013). 3D selection with freehand gesture. *Computers & Graphics*, 37(3), 101–120. <https://doi.org/10.1016/j.cag.2012.12.006>
- Rogers, Y., Sharp, H., & Preece, J. (2023). *Interaction design: Beyond human-computer interaction* (6th ed.). John Wiley & Sons Inc. <https://www.wiley.com/en-us/Interaction+Design+%3A+Beyond+Human+Computer+Interaction%2C+6th+Edition-p-9781119901099>
- Schmettow, M. (2012). Sample size in usability studies. *Communications of the ACM*, 55(4), 64–70. <https://doi.org/10.1145/2133806.2133824>
- Schneider, W., & Shiffrin, R. M. (1977). Controlled and automatic human information processing: I. Detection, search, and attention. *Psychological Review*, 84(1), 1–66. <https://doi.org/10.1037/0033-295X.84.1.1>
- Schramm, K., Gutwin, C., & Cockburn, A. (2016). Supporting transitions to expertise in hidden toolbars. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 4687–4698. <https://doi.org/10.1145/2858036.2858412>
- Shang, W., & Alena, K. (2025). Advancements in virtual reality technology: A systematic review of user experience and application trends. *Artificial Intelligence and Applications*, 3(4), 341–358. <https://doi.org/10.47852/BONVIEWAIA52024327>
- Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages. *Computer*, 16(8), 57–69. <https://doi.org/10.1109/MC.1983.1654471>
- Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2018). *Designing the user interface: Strategies for effective human-computer interaction* (6th ed.). Pearson Education. <https://www.jbe-platform.com/content/journals/10.1075/idj.17.2.14mar>
- Slater, M. (2018). Immersion and the illusion of presence in virtual reality. *British Journal of Psychology*, 109(3), 431–433. <https://doi.org/10.1111/bjop.12305>
- Sólfar Studios. (2017). *Everest VR on meta rift* [Accessed: 9 April 2024]. Reykjavik, Iceland. <https://www.meta.com/experiences/rift/1043021355789504/>
- Stillman, B. C. (2002). Making sense of proprioception: The meaning of proprioception, kinesthesia and related terms. *Physiotherapy*, 88(11), 667–676. [https://doi.org/10.1016/S0031-9406\(05\)60109-5](https://doi.org/10.1016/S0031-9406(05)60109-5)
- Taylor, J. L. (2009). Proprioception. In L. R. Squire (Ed.), *Encyclopedia of neuroscience* (pp. 1143–1149). Academic Press. <https://www.sciencedirect.com/science/article/pii/B9780080450469019070>

- Weijdom, J. (2022). Performative prototyping in collaborative mixed reality environments: An embodied design method for ideation and development in virtual reality. *Proceedings of the Sixteenth International Conference on Tangible, Embedded, and Embodied Interaction*, 1–13. <https://doi.org/10.1145/3490149.3501316>
- Wickens, C. D., Helton, W. S., Hollands, J. G., & Banbury, S. (2021). Memory and training. In *Engineering psychology and human performance* (5th ed., pp. 273–334). Routledge. <https://www.taylorfrancis.com/chapters/mono/10.4324/9781003177616-8/memory-training-christopher-wickens-william-helton-justin-hollands-simon-banbury>
- Wilson, A. D., & Golonka, S. (2013). Embodied cognition is not what you think it is. *Frontiers in Psychology*, 4. <https://doi.org/10.3389/fpsyg.2013.00058>
- Yan, Y., Yu, C., Ma, X., Huang, S., Iqbal, H., & Shi, Y. (2018). Eyes-free target acquisition in interaction space around the body for virtual reality. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 1–13. <https://doi.org/10.1145/3173574.3173616>
- Zhang, Z., Sarcevic, A., & Bossen, C. (2017). Constructing common information spaces across distributed emergency medical teams. *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 934–947. <https://doi.org/10.1145/2998181.2998328>

A INTERVIEW QUESTIONS

Participant Session

VE experience:

Navigation and layout

- Logical placement
- Working through each item
- Other Comments

SELECTION

In reach

- Were the menu items comfortably in reach?
- Did you use your dominant or non-dominant hand? Which ones did you use for certain tools?

Representation

- What indicated to you that you touched?
- Was the function of the tools obvious when you first saw them?
- What about after using them?
- Other

Visibility

- Was the state of the menu changing from menu to submenu clear?

Error recovery

- Was it easy or difficult to fix if you mistakenly selected something?
- Anything that makes you blame the system for the an incorrect selection

Easy to become familiar

- Did you find the menu system easy to become familiar with?
- Do you think your past experience with VEs (or lack thereof) contributed to you finding the menu system familiar/unfamiliar to you?

Comfort and fatigue

- Would you say that this menu system would cause more/less/equal amounts of fatigue compared to other VR interactions you've used before?

Eyes-off interaction

- Did you always look directly at the menu items when you were selecting them?
- Were there any menu items that you had to focus on more than others when selecting these menu items?

B FOCUS GROUP QUESTIONS






Focus Group

This menu system design draws inspiration from a utility belt. Utility belts provide quick access to useful tools without always looking.

1. Which part of the menu system did you enjoy using the most and why?
2. Some people pointed out that grabbing the measuring tape was difficult to perform.
 - (a) What are your opinions about this?
3. Some people found the vibrations to be annoying when they were just moving their hands around. It felt like they did something wrong or it just kept buzzing.
 - (a) How do you feel about this?
 - (b) *Extra*: buzzing due to gaming controller experience.
4. Some people would move the belt menu to a specific position and leave it there for the whole session. But others moved the belt menu more than once to adjust the positioning for specific subtasks. a. What are everyone's thoughts on this?
5. Some people would position the belt menu so that it is always visible. How do you feel about that (*Extra*: potential for HUD)?
6. Some people mentioned that the menu items would block their view of the size labels. a. What changes would you make to the menu system to deal with this issue?
7. The purpose of putting some menu items (e.g. measuring tape) out of direct eyesight was to see if you would try to select them without looking.
 - (a) Did any of you try to select menu items without looking at them?
 - (b) If so, what did you rely on to do so (e.g. vibrations, shadow, gesture/waving based on position remembered)?
 - (c) Did you try to apply the tools to the blocks without looking (e.g. painting the block)?
 - (d) Do you think that it would be easier to select from the various submenus if "dipping" was used instead (holding the block and dipping them into different buckets of paint)?

8. If there were one or more aspect(s) of the menu system that you could change,
 - (a) what would it be
 - (b) and why?
9. *Extra:* Some people mentioned that they would want to customize the placement of menu items.
 - (a) Is there anything else that you would want to be customizable?
 - (b) If so, how would you want it to be done?
10. Do you think that this kind of menus system is a feasible solution to be used in immersive 3D virtual environments?
11. *Extra:* Was there any aspect of the task (placing blocks correctly into boxes) that you struggled with that was not caused by the menu system.
12. *Extra:* For those who have experience with any other VR equipment (e.g. Oculus Rift), how transferable do you think this kind of menu system is to other VR equipment?
13. Question 5: Asking about HUD

Improving greybox fuzzing with dictionary-based mutations: A systematic literature review

Enock L. Dube^a , Boluwaji A. Akinnuwesi^a , Stephen G. Fashoto^{a,b} ,
Petros M. Mashwama^a , Vusi W. Tsabedze^a 

^a Department of Computer Science, University of Eswatini, Eswatini

^b Department of Informatics, Namibia University of Science and Technology, Windhoek, Namibia

ABSTRACT

Detecting deep bugs that are guided by complex conditions, based on specific byte sequences of the input, often requires input structure-aware or grammar-aware fuzzing strategies. However, the grammar or specification of the input may not be readily available. In this regard, there exists anecdotal evidence that dictionary-based mutations contribute to preserving the syntactic structure of input test cases and may approximate the efficacy of grammar-aware fuzzing. It is not yet clear as to which is the best strategy for automatically extracting fuzzing dictionary tokens from the codebase of the program under test. In this study we conduct a systematic review of the impact of dictionary-based mutations on the fuzzing process. We further review strategies for automatically extracting dictionary tokens and optimizing dictionary-based mutations. Our findings are that current strategies for extracting fuzzing dictionary are not optimised for highly structured input. Furthermore, about 58% of the reviewed state-of-the-art fuzzing tools rely on the random mutation operator distribution of respective baseline fuzzer. Moreover, the evaluation of these fuzzing tools report on aggregated performance of mutation operator scheduling algorithms, and not specific individual operators such as dictionary-based mutation operators.

Keywords Dictionary-based mutation, Mutational Fuzzing, Greybox fuzzing, Software vulnerability, Software testing

Categories • Software and its engineering ~ Software creation and management, Software verification and validation, Software defect analysis, Software testing and debugging

Email

Enock L. Dube – eldube@uniswa.sz (CORRESPONDING)
Boluwaji A. Akinnuwesi – bakinnuwesi@uneswa.ac.sz
Stephen G. Fashoto – sgfashoto@uniswa.sz
Petros M. Mashwama – petros@uniswa.sz
Vusi W. Tsabedze – vtsabedze@uneswa.ac.sz

Article history

Received: 4 March 2025
Accepted: 15 July 2025
Online: 22 December 2025

1 INTRODUCTION

Fuzz testing, also known as fuzzing, is a software testing technique that can be used to detect faults or weaknesses such as correctness bugs and security vulnerabilities in an input-parsing

Dube, E.L., et al. (2025). Improving greybox fuzzing with dictionary-based mutations: A systematic literature review. *South African Computer Journal* 37(2), 74–103. <https://doi.org/10.18489/sacj.v37i2.21430>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](https://creativecommons.org/licenses/by-nc/4.0/) 
SACJ is a publication of *SAICSIT*. ISSN 1015-7999 (print) ISSN 2313-7835 (online)

program (Liang et al., 2018). It works by repeatedly running the program under test (PUT) with mutated or fuzzed input test cases. Ever since its conceptualisation in 1990 (Miller et al., 1990), fuzz testing remains a widely used technique, and it provides an inexpensive mechanism for eliciting faulty program behaviour. Compared to other vulnerability testing techniques, such as static analysis and penetration testing (Halfond et al., 2011; Liu et al., 2012), fuzz testing scales well to large programs (Godefroid, 2020). However, a major limitation of most fuzzing frameworks is that they often fail to deal with programs that require highly structured input data such as JavaScript and XML objects (Zalewski, 2015). In this regard, fuzzing highly structured input often generates invalid input test cases that are promptly rejected in the parsing phase of the target program under test. In addition to model-based fuzzing approaches, such as grammar-based fuzzing, dictionary-based mutational fuzzing was introduced to mitigate the limitations of the grammar-blind nature of fuzzing (Zalewski, 2015).

Dictionary-based mutation is implemented by most of the state-of-the-art fuzzers such as American Fuzzy Lop (AFL) (Zalewski, 2013), libFuzzer (Serebryany, 2015) and Honggfuzz (Swiecki & Grobert, 2025). Despite anecdotal evidence (Zalewski, 2015) that dictionary-based mutation can approximate grammar-based input generation, and therefore improve efficiency of the fuzzing process, there exist a limited number of empirical studies on this approach. In this study, we conduct a systematic literature review on the impact of dictionary-based mutation on the fuzzing process. The significance of the study is that it provides a consolidated overview of dictionary-based mutational fuzzing. To the best of our knowledge, this is the first systematic literature review addressing this topic. The subsections that follow provide background information on the fuzz testing process and further discuss the dictionary-based mutation strategy in more detail.

1.1 Background of Fuzz Testing Process

Software vulnerabilities, such as memory corruption, remain a major cause of many severe threats to input parsing programs (Gan et al., 2018). Studies have shown that nation-state and independent hackers rely on the presence of software vulnerabilities to develop exploits that break down a target program execution with the intention of performing malicious actions such as control flow hijacking, information leakage, and denial of service attacks (Nagy & Hicks, 2019). Unlike independent hackers, nation-state hackers are sponsored by national governments to launch cyber-attacks that are typically aligned to political or military objectives. The target of these attacks could be an individual person, organisation or another state. Bugs and vulnerabilities in the software systems enable the execution of such attacks.

A software vulnerability is a defect that may originate from human mistake in the design of the software, and it may introduce an exploitable fault, flaw or bug in the code. A fault may result in a failure characterised by an observed deviation from expected behaviour of a program (IEEE, 1990). To identify and mitigate these vulnerabilities, software developers may employ various testing and vulnerability detection techniques, such as fuzz testing. The primary goal of this testing is to minimise the exploitability of a software system.

In principle, a vulnerability detection and analysis system should be sound and complete. A sound system never approves a vulnerable program (*no vulnerabilities are missed*) and is said to be complete if all secure programs can be approved (*no false vulnerabilities*). By extension, a vulnerability detection and analysis system is said to be both sound and complete if it can approve all secure programs and disapprove all vulnerable programs (*no missed vulnerabilities and no false vulnerabilities*) (Xie et al., 2005). In practice, however, most fuzz test techniques and strategies are neither sound nor complete (Ghaffarian & Shahriari, 2017). The goal of fuzz testing is to identify problematic program states which may be associated with security vulnerabilities (Serebryany, 2015). In recent years many fuzz testing techniques (King, 1976), tools (Böhme et al., 2019; Serebryany, 2015; Zalewski, 2013) and frameworks (Fioraldi, Maier et al., 2020) have been proposed, implemented and evaluated. These methods can be classified into three main strategic approaches: Blackbox, Greybox and Whitebox fuzzing. Figure 1 provides a workflow that summarises the common processing steps shared by these three different approaches.

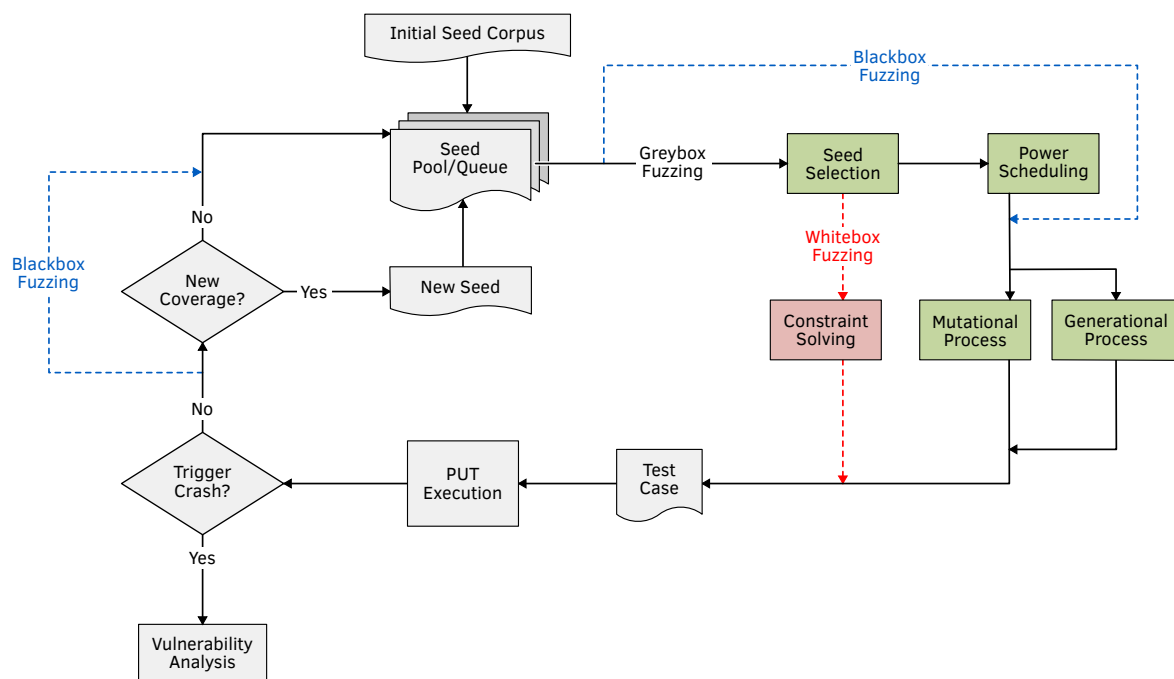


Figure 1: Workflow of Different Fuzzing Approaches^a

^a Adapted from P. Wang et al. (2024)

As depicted in Figure 1, the fuzzing process is a loop that commences by selecting a seed or input test case from a seed pool or queue. It uses a power schedule to determine the seed's energy. The energy specifies how many times each selected input may be mutated. The term power schedule refers to a mechanism that assigns more energy to interesting input test cases such as those that explore more code paths. The next step executes the program under test

using the newly generated input test case. The program execution is monitored to determine if it leads to a failure which must then be analysed to determine the presence of a bug or flaw in the code. As indicated in [Figure 1](#), some steps may be skipped depending on the fuzzing strategy adopted.

All three fuzzing strategic approaches repeatedly generate input test cases and execute the program under test while monitoring the program state to detect any abnormal behaviour caused by correctness bugs or security vulnerabilities in the code. Blackbox fuzzing (indicated by blue lines in [Figure 1](#)) randomly generates new input test cases and skips the input selection and power scheduling steps. In general, Blackbox fuzzing tests the specified behaviour of a target PUT and often does not require the source code (Zeller et al., [2024](#)). It is widely applicable without prior knowledge of internal program constructs. While random test case generation in Blackbox fuzzing is fast and scales well to large programs, its major limitation is that it may generate a lot of invalid input test cases. As a result, Blackbox fuzzing may take longer execution time to generate a new test case that triggers a failure in the target PUT. There is no guarantee that Blackbox fuzzing will identify all vulnerabilities and not raise false positives. In this regard, Blackbox fuzzing is neither sound nor complete (Ghaffarian & Shahriari, [2017](#)).

On the other hand, Whitebox fuzzing (indicated by the red lines in [Figure 1](#)) leverages program analysis techniques, such as symbolic execution (Cadar & Sen, [2013](#); King, [1976](#)) and constraint solving techniques (De Moura & Bjørner, [2008](#)), to generate new input test cases. Symbolic execution analyses a program and determines what inputs can explore each execution path in the code. Whereas Blackbox fuzzing tests the specified behaviour of the PUT, Whitebox fuzzing tests the implemented behaviour and often requires an analysis of the source code. When applied to small code components, such as unit testing, Whitebox fuzzing is guaranteed to generate input test cases that exercise all path conditions in the program under test, increasing the likelihood to identify bugs and detect software vulnerabilities (Böhme et al., [2019](#)). In this context, Whitebox fuzzing may be sound and complete. However, the symbolic execution and constraint solving techniques deployed in Whitebox fuzzing are computationally expensive and do not scale well to large programs (Godefroid et al., [2008](#)).

Greybox fuzzing combines aspects of both Blackbox and Whitebox fuzzing techniques and may work on both binary and open-source code. Whereas Blackbox fuzzing generates input test cases without any internal knowledge of the program under test, Greybox fuzzing leverages program instrumentation to generate feedback information, such as code coverage, which may be used to guide the input generation process. Additionally, Greybox fuzzing incorporates some aspects of Whitebox fuzzing, such as feedback information, which provides a Greybox fuzzer the ability to acquire some internal knowledge of the target PUT. However, Greybox fuzzing may avoid or limit the use of computationally expensive techniques such as symbolic execution. Similar to Blackbox fuzzing, Greybox fuzzing is neither sound nor complete. It provides no guarantee of identifying all vulnerabilities and may still raise false vulnerabilities (Ghaffarian & Shahriari, [2017](#)).

Greybox fuzzing approaches may be further divided into three categories: coverage-based

greybox fuzzing (CGF), directed greybox fuzzing (DGF) and regression greybox fuzzing (RGF). Whereas CGF seeks to cover all code paths in the target program under test, DGF focuses on reaching specific target locations, therefore reducing computational resource wastage on exploring unrelated code paths (Böhme et al., 2017). On the other hand, regression greybox fuzzing is an advanced technique that focuses on recently changed or frequently modified code segments (X. Zhu & Böhme, 2021). Directed greybox fuzzing has gained prominence as an efficient approach for targeted software testing in specific scenarios such as patch testing and bug reproduction (P. Wang et al., 2024). Most of the state-of-the-art fuzzing tools are coverage-based greybox fuzzers. Examples in this category include AFL (Zalewski, 2013), AFLFast (Böhme et al., 2019), AFL++ (Fioraldi, Maier et al., 2020), ATTUZ (S. Zhu et al., 2024), and FIRM-COV (Kim et al., 2021). Directed greybox fuzzing tools include ALFGo (Böhme et al., 2017), GTFuzz (Li et al., 2020), and Hawkeye (H. Chen et al., 2018). Examples of regression greybox fuzzers include AFLCHURN (X. Zhu & Böhme, 2021). Greybox and Whitebox fuzzing may be combined to build hybrid fuzzing tools that leverage features of both approaches. Examples of hybrid fuzzers include Driller (Stephens et al., 2016), Vuzzer (Rawat et al., 2017), Angora (P. Chen & Chen, 2018) and Savior (Y. Chen et al., 2020).

The following section introduces a working example that is used to elaborate on the different fuzzing approaches and further clarifies some of the terminology used in this study.

1.2 Working Example: PNG Parser

Listing 1 shows sample code that may be used to parse and process a Portable Network Graphics (PNG) binary file. A PNG file starts with an 8-byte signature that identifies the file as containing a PNG image, followed by sequence of chunks or sections.

The PNG signature consists of the following hexadecimal byte sequence: `0x89 0x50 0x4E 0x47 0x0D 0x0A 0x1A 0x0A`. The hexadecimal value `0x89` refers to the ordinal value of the per mille symbol (‰) in the American Standard Code for Information Interchange (ASCII) character set. The values `0x50`, `0x4E` and `0x47` are the ordinal values for character symbols *P*, *N* and *G*. Similarly, `0x0D` and `0x0A` are values for carriage return (*CR*) and linefeed (*LF*) control characters respectively. The value `0x1A` represents the substitute (*SUB*) control character. The values in the PNG signature are examples of magic bytes. The term magic byte or magic value refers to specific bytes of the input sequence that must be matched to access certain code paths in an input parsing program. As shown in **Listing 1**, a program that processes a PNG binary file contains conditional statements that verify that the file contains the appropriate signature and is not corrupted. Ideally, a PNG parsing program, such as the *PNG_parser* function in **Listing 1**, may only process a PNG file with a valid signature.

```
#define MAX_SIZE (16 * 1024 * 1024) // 16 megabytes

bool check_png_signature(char input_buffer[], int num_bytes_to_check)
{
    // PNG signature magic values
    const char png_signature[8] = {
        0x89, 0x50, 0x4E, 0x47,
```

```

    0x0D, 0x0A, 0x1A, 0x0A
};

// C1: limit to bytes [1..8]
if ((num_bytes_to_check < 1) || (num_bytes_to_check > 8)) {
    return false;
}
else
// C2: check hard-coded PNG signature magic values
    return (memcmp(input_buffer, png_signature, num_bytes_to_check) == 0);
}

void PNG_parser(const char *png_input_file, int buffer_size)
{
    char *input_buffer = (char *)malloc(buffer_size);
    if (input_buffer == NULL) {
        // TODO: handle allocation error
        return;
    }

    // ...

    FILE *infile = fopen(png_input_file, "rb");
    // Handle file open failure
    // ...

    int size = fread(input_buffer, 1, buffer_size, infile);

    // C3: check magic bytes
    if (check_png_signature(input_buffer, 8) == true) {

        int chunk_start_pos = 8; // chunk start position; after signature
        // BUG1: buffer over-read if input_buffer < 8 bytes → validate size first

        while (chunk_start_pos < size) {

            char chunk_lenbuffer[4];
            memcpy(chunk_lenbuffer, input_buffer + chunk_start_pos, 4);

            int chunk_length = get_big_endian(chunk_lenbuffer);

            char chunkbuf[5];
            int chunk_type_start_pos = chunk_start_pos + 4;

            // BUG2: buffer over-read if input_buffer < 13 bytes → validate first
            for (int index = 0; index < 4; index++) {
                chunkbuf[index] = input_buffer[chunk_type_start_pos + index];
            }

            chunkbuf[4] = '\0';

            // process chunk here
            // ...

        }

    }

    // close file and free buffer
    // ...
}

```

Listing 1: Sample code to parse a PNG file

The PNG signature is followed by a sequence of chunks/sections. Each chunk is indicated by a chunk code or type such as *IHDR*, *IEND*, *IDAT*, and *PLTE*. The codes are 4-byte sequences consisting of hexadecimal representation of each character symbol in the code name. For instance, the hexadecimal values of characters *I*, *D*, *A*, *T* in the *IDAT* code. The chunk types/codes may also be considered magic bytes or values that are often used in complex conditional statements and therefore, guard the execution of certain blocks of code in a PNG file parsing program.

A PNG file begins with an *IHDR* chunk and ends with an *IEND* chunk. Between *IHDR* and *IEND* other chunk types, such as *IDAT* and *PLTE*, may appear multiple times. Figure 2 shows the format of each chunk.

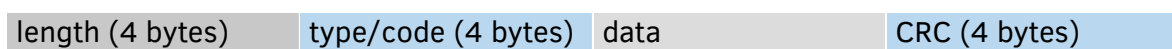


Figure 2: PNG chunk format

Each chunk starts with a 4-byte length field which specifies the size of the PNG image. The length field is followed by a 4-byte type field (for instance *IHDR*, a data field, and a 4-byte Cyclic Redundancy Check (CRC)). The value of the length field depends on the size of the image represented in the data field. The CRC is a checksum value that is calculated based on byte sequences in the chunk type field and data field. In this regard, when parsing a PNG file, the CRC may be used to validate that the data is not corrupted. The RFC 2083 PNG specification (Boutell, 1997) contains a more detailed discussion of the PNG file format.

The *PNG_parser* function in Listing 1 attempts to first read and validate the PNG signature, and then proceeds to extract information from the collection of chunks. The code contains buffer over-read bugs or vulnerabilities (indicated as *BUG1* and *BUG2* in Listing 1). A buffer over-read is a type of vulnerability that occurs when code execution tries to read data beyond the end of a buffer in memory.

The *check_png_signature* function in Listing 1 takes an input buffer, representing a PNG file content, and validates the signature. Since the signature refers to the first 8 bytes of the input buffer, the first conditional statement (*C1*) limits the values for the second argument, *num_bytes_to_check*, to between 1 and 8.

In the context of Blackbox fuzzing, a randomly generated value for *num_bytes_to_check* will frequently generate a lot of values that are either greater than 8 or less than 1. In other words, the probability of randomly generating a 4-byte integer value between 1 and 8 is small ($8/2^{31}$). In this regard, the *check_png_signature* function often evaluates to false, and code execution fails to reach the validation code in conditional statement *C2*. Blackbox fuzzing will also take a long time to randomly generate an input that directs code execution to go beyond conditional statement *C3* in the *PNG_parser* function. Inevitably, it will take a longer time to identify the bugs or vulnerabilities (*BUG1* and *BUG2*) guarded by *C3*. The conditional statements *C1*, *C2* and *C3* are examples of sanity check. The term sanity check refers to security measures that are used to verify the validity of the input and help ensure that the code processes it correctly.

On the other hand, a Whitebox fuzzing approach may use symbolic execution to reach and explore the vulnerable code segments much faster than Blackbox fuzzing. A symbolic execution of the *check_png_signature* function identifies a symbolic value, such as a_1 , and assigns it to variable *num_bytes_to_check*. Upon reaching the conditional statement *C1*, it would evaluate the path constraint: $(a_1 < 1) \vee (a_1 > 8)$. At this point of the code execution, a_1 could take any value, and symbolic execution can fork into two paths and proceed along both branches of the if-else conditional statement *C1*. Each path gets assigned a copy of the program state at the branch instruction as well as a path constraint. In this example, the path constraint is $(a_1 < 1) \vee (a_1 > 8)$ for the *IF* branch and its negation, $(a_1 \geq 1) \wedge (a_1 \leq 8)$, for the *ELSE* branch. Both paths can be symbolically executed independently of one another. When each path's execution terminates, symbolic execution computes a concrete value for a_1 by solving the accumulated path constraints on each path. These concrete values can be thought of as concrete input test cases. In this example, the constraint solver would determine that in order to reach the else-branch of conditional statement *C1*, a_1 should be a value between 1 and 8. This branch executes the validation code in conditional statement *C2*. In this regard, Whitebox fuzzing generates valid input test cases significantly faster than Blackbox fuzzing and is often able to systematically explore the vulnerable code segments (*BUG1* and *BUG2*) guarded by conditional statement *C3*. However, for large programs with nested conditional statements, the path constraints may increase in length and complexity, making it difficult for a constraint solver to resolve.

1.3 General Weaknesses of Blackbox, Greybox, and Whitebox Fuzzing

The general weaknesses of Blackbox, Greybox, and Whitebox fuzzing lie in their limitations regarding scalability, accuracy, and computational cost. Blackbox fuzzing, while fast and scalable to large programs, suffers from inefficiency in identifying bugs, as it often generates a high volume of invalid input test cases. As illustrated in Section 1.2, randomly generated input test cases often fail sanity checks. It may take a lot of trials to generate a test case that reaches and executes the vulnerable code segments. This may contribute to a slower rate of vulnerability detection. Whitebox fuzzing, which is exhaustive in exploring program paths through symbolic execution, faces scalability issues in larger programs that contain complex path constraints. The constraints solver may fail to resolve the path constraints. Greybox fuzzing represents a middle ground between Blackbox and Whitebox fuzzing and uses code coverage feedback to guide input test case generation. However, Greybox fuzzing is prone to missing certain vulnerabilities and may raise false alarms, as it lacks the completeness of Whitebox fuzzing.

1.4 Mutational vs Generational Fuzzing

Mutational or mutation-based fuzzing treats program inputs as a sequence or array of bytes, and repeatedly applies byte-level mutation operators to generate new input test cases. These

include bitflip & byteflip operators, arithmetic increment & decrement operators, byte insertion, deletion & overwrite operators, and dictionary-based mutation operators. These operators characterise which bytes in the input byte-array to mutate and how to mutate them. For instance, the *check_png_signature* in Listing 1 takes two arguments: An *input_buffer* array and *num_bytes_to_check* integer value. As shown in Figure 3, mutational fuzzing views these two values as a single sequence or array of bytes. In this view, the *num_bytes_to_check* bytes are concatenated at the end of the *input_buffer* bytes, to form one byte sequence.

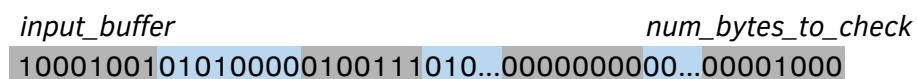


Figure 3: Sample representation of input as sequence of bytes

Mutational fuzzing repeatedly applies mutation operators to the input byte sequence. In most mutational fuzzing tools such as AFL (Zalewski, 2013), the fuzzing process begins with a deterministic stage followed by a havoc (non-deterministic) stage. The deterministic stage is executed only once for each seed input. It starts at the beginning of each seed input byte sequence, sequentially selects each byte and applies each available mutation operator. The deterministic stage may also be used to assign initial energy to each seed in the corpus. On the other hand, the havoc stage randomly selects a mutation operator and applies it to a randomly selected byte location in the input byte sequence. Some fuzzers, such as libFuzzer (Serebryany, 2015) do not implement the deterministic stage and are often referred to as non-deterministic fuzzers. However, a majority of mutational fuzzers implement both stages and apply byte-level mutations to generate new input test cases.

Byte-level mutations are often syntax blind and therefore may generate many invalid input test cases. For instance, randomly mutating the *input_buffer* bytes (in Figure 3) may alter the signature resulting in the file content not recognisable as a PNG image. For instance, a single bitflip operation on the second byte in Figure 3 may change it from 01010000 to 01010001 resulting in a signature that contains the character sequence QNG instead of PNG. This operation generates an invalid input that is rejected by the parsing process since it fails the validation check. In this regard, whereas byte-level mutations may be successful for simple formats, they generate a high number of invalid test cases for highly structured input formats such as PNG files. As another example, randomly mutating the *input_buffer* (in Figure 3) may distort the structure of the PNG chunks. For instance, a number of bitflip operations that changes the byte representation for character *R* in the *IHDR* chunk code from 01010010 to 00100011 will mutate the chunk type/code to be *IHDR*#. Since the CRC checksum is calculated based on the chunk type field and chunk data field, this mutation results in a corrupted *IHDR* chunk with an incorrect CRC checksum. In this example, unless the *PNG_parser* code has sufficient sanity checks to validate the syntactic structure of the PNG chunks, it may fail to correctly recognise *IHDR* header chunk or may compute the wrong checksum. For a target parsing program with strong sanity checks, many of the syntactically invalid inputs may be rejected or may fail to explore and test the vulnerable code segments. To mitigate this challenge, a syntax-aware

approach such as generational or grammar-based fuzzing may be used.

Generational or generation-based fuzzing (Eberlein et al., 2020; Godefroid et al., 2008), sometimes referred to as grammar-based fuzzing, generates input test cases from an input language specification such as grammar or a set of rules. Its main advantage is that it guarantees that all generated test cases are syntactically valid hence the fuzzing process does not waste computational resources parsing invalid inputs. However, since most programs lack a formal grammar of well-formed inputs, mutational fuzzing techniques are widely adopted.

Most fuzzing tools adopt mutation-based strategies and repeatedly apply syntax-blind byte-level mutations. As already noted, mutational fuzzing may generate a high number of invalid test cases. To mitigate this challenge, dictionary-based mutation was introduced to make-up for the grammar-blind nature of AFL (Zalewski, 2015). Subsequently, dictionary-based mutation has been implemented in other state-of-the-art fuzzers such as libFuzzer (Serebryany, 2015) and Honggfuzz (Swiecki & Grobert, 2025). In Greybox fuzzing, the feedback provided by the compile-time instrumentation makes it possible to identify syntax tokens in some types of files, and further detect that certain combinations of terms constitute a valid grammar of the input test cases. In this regard, fuzzing dictionaries have been shown to enable the fuzzer to rapidly reconstruct the grammar of highly verbose languages such as XML, SQL and JavaScript (Zalewski, 2015).

A fuzzing dictionary is a text file that contains a collection of commonly occurring keywords, strings, interesting byte sequences and magic bytes/values. As discussed in the working example in Section 1.2, magic bytes such as the PNG signature, `0x89 0x50 0x4E 0x47 0x0D 0x0A 0x1A 0x0A` and chunk codes (*IHDR*, *IEND*, *IDAT*, *PLTE*), may be extracted to a fuzzing dictionary. The Google Fuzzing Dictionaries (Google, 2025) repository provides examples of fuzzing dictionary for different file formats.

1.5 Dictionary-Based Mutation

A fuzzing dictionary consists of a list of basic syntax tokens. These tokens may be manually extracted or automatically identified during the fuzzing process. Most fuzzing frameworks implement two types of dictionary-based mutation operators: *insert* and *overwrite*. Whereas other fuzzing tools may use different naming conventions, in AFL these operators are called *user extras* operator and *auto extras* operator respectively. The AFL *user extras* operator selects a token from a fuzzing dictionary and inserts its bytes into the input test case. The AFL *auto extras* operator overwrites bytes in the input test case with a dictionary token recognized and automatically extracted by AFL during deterministic stage. Other coverage-based mutational fuzzers such as libFuzzer and Honggfuzz implement similar mechanisms for dictionary-based mutation. While AFL performs dictionary-based operations during the deterministic stage as well-as havoc stage of fuzzing, libFuzzer is non-deterministic and performs dictionary-based mutation only during the random havoc stage. The general pseudocode for dictionary-based mutation is shown in Algorithm 1 which is adapted from Shastry et al. (2017).


```

function DICTIONARY_FUZZ(input_bytes, dictionary, deterministic)

    dict_token ← selectRandomTokenFrom(dictionary)

    if deterministic = TRUE then
        for each offset in input_bytes do
            FUZZ_TOKEN_OFFSET(input_bytes, dict_token, offset)
        end for
    else
        offset ← selectRandomOffsetPosition(sizeof(input_bytes))
        FUZZ_TOKEN_OFFSET(input_bytes, dict_token, offset)
    end if

end function

function FUZZ_TOKEN_OFFSET(input_bytes, dict_token, offset)

    overWriteFromOffsetPosition(input_bytes, offset, dict_token)
    runProgramUnderTest(program, input_bytes)

    insertTokenAtOffsetPosition(input_bytes, offset, dict_token)
    runProgramUnderTest(program, input_bytes)

end function

```

Algorithm 1: Pseudocode for Dictionary-based Mutation

Algorithm 1 is implemented in most Greybox fuzzers such as AFL, Honggfuzz and libFuzzer. During dictionary-based mutation, a token is selected from the dictionary and inserted between bytes or written over byte sequences of the same length. Whereas such dictionary-based mutations can generate syntactically valid input test cases, they can also destroy the syntax structure of the input tests case. In this regard, a study (J. Wang et al., 2019) conducted in 2019 proposed an enhanced dictionary-based algorithm that identifies an ideal location to insert or overwrite bytes sequences and therefore preserve syntax structure. The results of the study provide evidence that dictionary-based mutations help improve validity of generated input test cases, resulting in improved performance of fuzz testing. The performance of fuzzing may be measured in terms of effectiveness and efficiency. The term effectiveness refers to the total number of vulnerabilities discovered, and the term efficiency refers to the rate at which vulnerabilities are discovered. Whereas counting the number of discovered vulnerabilities is an ideal metric for fuzzing effectiveness, it is not always feasible. For instance, a fuzz testing run that does not discover any unknown bugs or vulnerabilities is not necessarily ineffective. As a result, proxy metrics such as path or code coverage are often used as approximate measures of effectiveness. That is, a fuzzing technique that explores a high proportion of code execution paths is more likely to a discover a bug or vulnerability and therefore could be considered highly effective. Similarly, low code coverage reduces the likelihood of bug or vulnerability discovery.

The focus of this study is on dictionary-based mutation as a targeted approach to enhance fuzz testing efficiency and effectiveness. First, we interrogate the evidence that dictionary-

based mutation has an impact on the fuzzing process. We further investigate the impact of mutation operator selection and scheduling. The study objective and research questions are presented in [Section 1.6](#).

1.6 Objective and Research Questions

The primary objective of this study is to investigate the role and impact of dictionary-based mutations in generating valid input test cases for fuzz testing. It reports on strategies and techniques that have been employed in previous studies to extract fuzzing dictionary tokens from the codebase of the program under test. Furthermore, the review examines how dictionary-based mutations have been used to improve the effectiveness and efficiency of the fuzzing process in the context of other optimisation techniques such as mutation operator selection and scheduling. In order to achieve this objective, the review process is guided by the following research questions (RQ):

RQ1: How does dictionary-based mutation impact fuzz testing effectiveness and efficiency?

RQ2: What approaches and techniques can be used to extract fuzzing dictionary tokens from the codebase of the program under test?

RQ3: What are the limitations of dictionary-based mutations in fuzz testing?

RQ4: What strategies can be used to optimize dictionary-based mutation operator selection and scheduling?

RQ5: How can alternative augmentation strategies be deployed to resolve the limitations of dictionary-based mutations?

We chose **RQ1** to examine the impact of dictionary-based mutations on the fuzz testing process. **RQ2** expands from **RQ1** and focuses on different approaches, and the techniques, which have been applied to extract fuzzing dictionary tokens. **RQ3** seeks to identify limitations of dictionary-based fuzzing and articulate research gaps. In **RQ4** we investigate strategies that have been used to optimize dictionary mutation operator selection and scheduling. We further critically analyse how dictionary-based mutations may be scheduled more optimally to positively impact the fuzzing efficiency and effectiveness. **RQ5** seeks to explore alternative strategies that may be deployed to address the limitations of dictionary-based mutation.

The rest of the paper is structured as follows: [Section 2](#) presents the research methodology. A discussion of the results follows in [Section 3](#), and we present future works and conclusion in [Sections 4](#) and [5](#) respectively.

2 METHODOLOGY

A Systematic Literature Review (SLR) was conducted following the guidelines proposed by Kitchenham (2007). It is guided by the objective and the research questions. The SLR processes are discussed in this section. Furthermore, this section details the search strategy along with the inclusion and exclusion criteria used to identify the relevant literature.

2.1 Search Query

In the systematic review of literature, relevant primary studies were extracted based on the following search terms: **fuzzing**, **fuzz testing**, **mutation**, and **dictionary**. We combined the search terms to create the following search string: **(“fuzzing” OR “fuzz testing”) AND “mutation” AND “dictionary”**. Inclusion and exclusion criteria were used to filter the query results.

2.2 Information Sources

The primary sources used for the literature search included the following databases: Association of Computing Machinery (ACM) Digital Library, IEEE Explore and ScienceDirect. The Google scholar database search engine was used to find additional publications from proceedings of top-level conferences on security and software engineering such as the USENIX Security Symposium, and the Network and Distributed System Security Symposium (NDSS). It is worth noting that a significant number of publications on fuzz testing are discussed in the USENIX Security symposium.

2.3 Inclusion and Exclusion Criteria

The main inclusion criterion is that only peer reviewed publications are considered for this review. In addition, the publication must be written in English, its content must cover the scope of the search terms and must have been published between the years 2010 and 2024. The year range is based on the observation that a significant large number of publications on fuzz testing appeared during that period. In the same period, current state-of-the-art fuzzing tools such as AFL and libFuzzer were developed. Furthermore, only primary studies appearing in journals and conference papers were included in this review. Therefore, book chapters, literature review papers, editorials, comments, conference keynotes and short papers (less than 4 pages) were all excluded.

2.4 Screening of Publications

In addition to the inclusion and exclusion criteria, each paper publication was screened based on the title and abstract to determine its relevance with regards to the objective of the literature review. Publications with irrelevant content were removed. The focus was on application

software file-based fuzzing publications, as opposed to other fuzzing domains such as kernel fuzzing, network protocol fuzzing and parametric fuzzing. In this regard, we excluded publications focusing on fuzzing interfaces on hardware, central processing unit (CPU), embedded devices, Internet-of-Things (IoT), Industrial control systems, Android systems, kernels and network protocols.

2.5 Assessment of Literature search

The quality assessment criteria (QAC) that were used to screen each publication is shown in Table 1. It is based on the publication, methodology and whether it addresses specific aspects of dictionary-assisted mutational fuzzing. The overall design of the assessment form was adapted from a study conducted by Raharjana et al. (2021) by formulating specific questions appropriate for this review.

Table 1: Quality Assessment Form

Item	Quality Assessment Criteria (QAC)	Score and Description
QAC1	Is the goal of the research study clearly stated?	-1: NO, 0: Partially, 1: YES
QAC2	Is the research methodology described in detail?	-1: NO, 0: Partially, 1: YES
QAC3	Does the study address specific aspects of dictionary-based mutation?	-1: NO, 0: Partially, 1: YES
QAC4	Is the proposed dictionary-based mutation strategy or intervention proven to work or sufficiently evaluated?	-1: NO, 0: Partially, 1: YES
QAC5	Do other scholarly publications reference the study?	-1: NO, 0: Partially, 1: YES

A Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) (Page et al., 2021) flow diagram was used to summarise the results of the screening process and assessment criteria, as shown in Figure 4.

Based on the inclusion and exclusion criteria, the database search engines (ACM Digital Library, IEEE Xplore, and ScienceDirect) were configured to retrieve only Journal and conference papers. The initial ACM Digital Library search resulted in 203 papers, the IEEE Explore search resulted in 211 papers, and ScienceDirect search resulted in 31 papers. The papers were read in more detail, and some were removed because the content did not match the main objective and scope of this review. For instance, some publications did not address any aspect of dictionary-based mutation.

Of the retrieved studies, twenty-five (25) met the inclusion criteria and were judged suitable to address the research questions. Each of these publications articulated a clear research objective, described the methodology in detail, and were cited in subsequent scholarly literature publications. Seven of the twenty-five studies specifically investigated methods for extracting a fuzzing dictionary from a target program's codebase.

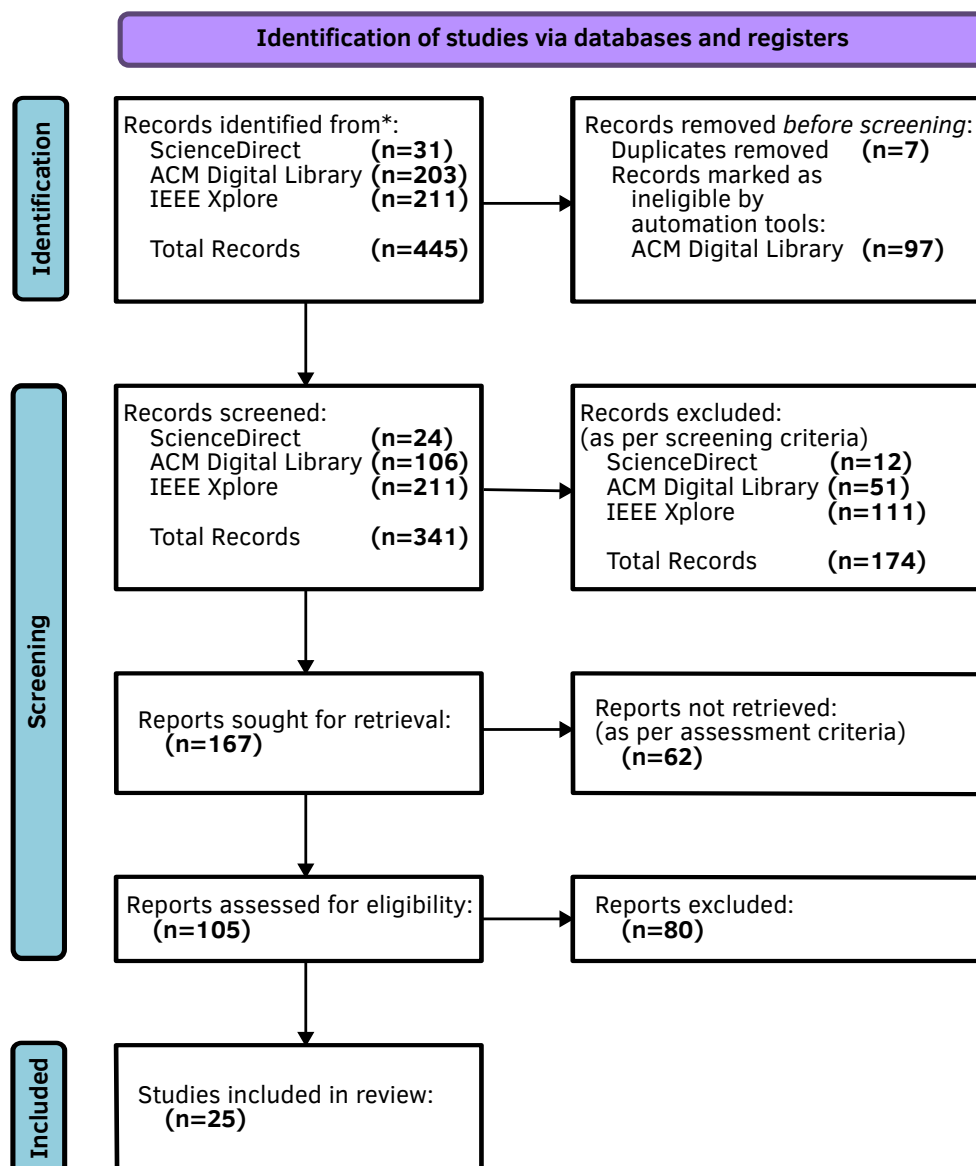


Figure 4: PRISMA Flow Diagram for Literature Search Results

2.6 Literature Data Extraction and Synthesis

To facilitate a synthesis of the publications, literature data was extracted and summarised using a data extraction form that summarises information appropriate to answering each of the review questions. For each reviewed publication, we recorded information such as the research focus, method, design, evaluation metric and limitations identified.

3 RESULTS AND DISCUSSION

Using the research questions (RQ) as a guide, and based on data extracted in [Section 2.6](#), this section provides an analysis and synthesis of the identified publications.

3.1 Impact of dictionary-based mutation on fuzz testing effectiveness and efficiency (RQ1)

State of the art fuzzing tools such as AFL (Zalewski, [2015](#)) have provided anecdotal evidence that user defined fuzzing dictionaries can improve code coverage. The implementation of both AFL and libFuzzer provide a collection of predefined general-purpose fuzzing dictionaries. However, these dictionaries are manually extracted, by application domain experts, from the codebase of the program under test. The Google Dictionaries project (Google, [2025](#)) also maintains a collection of general-purpose fuzzing dictionaries and makes them available as open source. A few studies have evaluated the impact of target specific fuzzing dictionaries that are automatically extracted from the codebase of the program under test (Ebrahim et al., [2022](#); Shastry et al., [2017](#); Wu et al., [2025](#)). An evaluation of results from these studies has shown that target specific fuzzing dictionaries are more effective than general-purpose dictionaries.

The FuzzingDriver (Ebrahim et al., [2022](#)) is one of the latest frameworks proposed to automatically extract fuzzing dictionary tokens. It can be executed before the fuzzing process runs, and therefore, does not add any computational overhead to the fuzzing process. The FuzzingDriver is generic and therefore, may be used to extract dictionary tokens from any target program. An evaluation of the FuzzingDriver extracted dictionaries, against general-purpose Google fuzzing dictionaries (Google, [2025](#)), showed improved code coverage. According to the developers of the FuzzingDriver framework, the efficacy of the FuzzingDriver on bug coverage is yet to be assessed.

On the same note, the Customized Dictionary Fuzzing (CDFUZZ) (Wu et al., [2025](#)) tool evaluated the impact of target specific dictionaries extracted using the FuzzingDriver. The study concluded that the dictionary-based mutation strategy improved fuzzing effectiveness and performed better than other fuzzing exploration strategies such as input-to-state correspondence proposed in REDQUEEN (Aschermann et al., [2019](#)) fuzzing tool, the QSYM (Yun et al., [2018](#)) SMT-solver based concolic engine, and gradient-based algorithm proposed in Angora (P. Chen & Chen, [2018](#)).

The Orthrus framework (Shastry et al., [2017](#)) proposed a mechanism that uses static compile-time analysis to extract fuzzing dictionary tokens. On evaluation of tcpdump network protocol, the Orthrus dictionary showed up to 10% improvement on code coverage compared to baseline AFL, and up to 8% improvement compared to baseline AFLFast (Böhme et al., [2019](#)). Similarly, an evaluation on Deep Network Traffic Inspection (nDPI) showed up to 15% and 5% improvement compared to baseline AFL and AFLFast respectively. The results of the evaluation also showed improved bug coverage. Although Orthrus was originally tested

only on network protocol messages, the authors indicate that its design could be extended to support file format parser applications such as PNG parsers.

In a study conducted in 2020, Mathis et al. proposed a fuzzing tool called LFuzzer that extends Dynamic Taint Analysis (DTA) to not only automatically infer fuzzing tokens, but also generate seed input test cases. DTA is a data flow tracking technique that is widely used for vulnerability analysis. It tracks tainted data during program execution and therefore, provides insight into how data flows through the program at runtime. The set of tokens inferred while using LFuzzer can be added to a fuzzing dictionary. The study also evaluated the LFuzzer extracted dictionary on highly structured input formats such as JSON and LISP. The findings indicate that, on average, the proposed approach achieves up to 17% more code coverage compared to baseline AFL.

Another study (Bundt et al., 2021) was conducted to explore the difference between fuzzing synthetic bugs and real-world bugs. The study concluded that coverage-guided fuzzers can effectively discover synthetic bugs in a LAVA (Dolan-Gavitt et al., 2016) generated synthetic dataset with techniques such as dictionary-based mutation and comparison splitting. The study utilised a dictionary of constant values parsed from a dis-assembly of the target program.

In 2021 Metzman et al. proposed and implemented a benchmarking platform and evaluated several fuzzing techniques. On the aspects of fuzzing dictionary, the results reflected minor differences between fuzzing with or without dictionary-based mutation enabled.

As highlighted in this section, there exist a limited number of research studies that provide empirical evidence of the impact of dictionary-based mutation on the efficiency and effectiveness of the fuzzing process. Moreover, a few studies provide details of how to automatically extract dictionary tokens from the codebase of the program under test. Section 3.2 looks at different strategies that have been used to extract fuzzing dictionary tokens.

3.2 Strategies for extracting fuzzing dictionary tokens from the program under test code base (RQ2)

The task of generating a dictionary from the codebase of the target PUT is strenuous and time consuming (Ebrahim et al., 2022). As a result, a lot of fuzzing tools and frameworks use predefined general-purpose dictionaries such as those provided by Google Fuzzing Dictionaries (Google, 2025). Furthermore, the source code release of AFL contains generic dictionaries for many common file formats which can be used in fuzzing runs. Moreover libFuzzer and Honggfuzz support AFL dictionaries and OSS-fuzz (Ding & Goues, 2021) integrates dictionaries in some of its fuzzing projects. While these generic dictionaries may improve code coverage, they lack target-specific information, and the impact is not optimised (Ebrahim et al., 2022). Additionally, manual extraction of a fuzzing dictionary requires expert knowledge. Therefore, in real world applications, there is a need to automatically generate target specific fuzzing dictionaries. Table 2 summarises the different approaches used to extract dictionary tokens.

The most basic strategy to create a fuzzing dictionary is to scrape through the source code of the target PUT and extract constants and string literals to a fuzzing dictionary. Some stud-

Table 2: Summary of Token Extraction Methods

EM# ^a	Method / Strategy	Fuzzing Tool	S/B ^b
EM_01	Domain expert manually extracts dictionary tokens	AFL libFuzzer	S
EM_02	Use byte and string comparison to identify tokens	AFL	B
EM_03	Auto-detect tokens during deterministic bitflip stage by looking for groups of bits that always produce the same coverage even after mutation	AFL	B
EM_04	Scrape the application binary code to create a dictionary	AFL & Thompson Sampling Deep Reinforcement Learning CONFETTI	B
EM_05	Syntactically extract input fragments from program abstract syntax tree (AST) and semantically extract conjunctions of input fragments from the program control flow graph (CFG) by statically analysing program data flow and control flows.	Orthrus	S
EM_06	Use CodeQL language queries to extract valuable information such as commonly occurring keywords, strings, and arguments of comparison functions	FuzzingDriver	S
EM_07	Use context sensitive data-flow analysis; leverage Guard Tokens; Extract strings from comparison functions and variable definitions	GTFuzz	S
EM_08	Auto-tokens – extracts tokens from instructions and comparison function	AFL++ libAFL	S
EM_09	Extract tokens from comparison instructions and functions with immediate values	libAFL	B
EM_10	Extracts all strings from binary codebase and selects strings used as arguments to library API and add to dictionary	FIRM-COV	B
EM_11	Learning input tokens; Explores branches of lexical analysis and extract dictionary tokens;	LFuzzer	B
EM_12	Uses extended dynamic tainting of implicit data transformation to produce/infer dictionary tokens and seed input test cases.	LFuzzer	B
EM_13	Leverage Input-to-state(I2S) correspondence and add all values that contain non-zero or non-0xff bytes to a specific fuzzing dictionary	FairFuzz REDQUEEN	B
EM_14	During compilation, extract all constant string comparison function parameters to a file; Need link-time optimisation (LTO) to pass auto-tokens with no overhead;	AFL++ libAFL	S
EM_15	Compile subject into human readable bit code format, and then extract string literals by iterating over the global values of the bit code file and writing the global strings to fuzzing dictionary	LFuzzer	B
EM_16	Use Linux strings tool to extract printable strings from a binary file and use output as dictionary	Deployed in older fuzzing frameworks	B
EM_17	Using ANGR binary analysis framework to help fuzzer identify possible magic values in the program under test	T-Fuzz	B
EM_18	Using hexadecimal editors, such as 010Editor, to extract interesting byte sequences (tokens) from input file format	AFLSmart	B

[continued ...]

^a EM = Extraction Method^b Source / Binary

Table 2: Continued...

EM#	Method / Strategy	Fuzzing Tool	S/B
EM_19	Use <code>AFL_LLVM_DICT2FILE=/absolute/path/file.txt</code> and during compilation all constant string compare parameters will be written to this file and later used as fuzzing dictionary.	AFL++	S
EM_20	Using <code>clang-sdict</code> that performs a front-end pass on source code collecting constant string tokens used in potentially data-dependent control flow.	Statistical Evaluation	S
EM_21	Use CodeQL to extract dictionary for each seed instead of overall dictionary from all seeds.	CDFUZZ	S

ies (Böttinger et al., 2018; Karamcheti et al., 2018; Kukucka et al., 2022) deployed this strategy. The dictionary generated in this way may contain less useful strings such as comments in the source code. In cases where source code is not available, another approach is to scrape dictionary tokens from the application binary code (Kukucka et al., 2022). In addition to manual methods, strategies for extracting fuzzing dictionary tokens range from scraping constant strings from input files, to using binary analysis frameworks such as ANGR¹, 010Editor², and Linux strings³ tool.

Auto-token detection strategies are used in fuzzers such as AFL, libFuzzer, AFL++ (Fioraldi, Maier et al., 2020) and libAFL (Fioraldi et al., 2022). Whereas AFL auto-detects tokens during deterministic bitflip mutation, other fuzzers auto-detect tokens from instructions and comparison functions (Fioraldi, Maier et al., 2020).

In an experimental framework called Orthrus, Shastry et al. (2017) used static syntactic analysis to automatically identify input fragments from the target program abstract syntax trees (AST). These input fragments can be added to a fuzzing dictionary. Furthermore, the framework used static semantic analysis to identify conjunctions of input fragments from the target PUT control flow graph (CFG). These were also added to the fuzzing dictionary.

A study conducted by Kim et al. (2021) outlined a detailed algorithm that first extracts all readable string constants from the data section of the binary input file, then identifies and constructs a list of addresses of the extracted strings. This is followed by a series of steps that include reference mining (identifying instructions that reference a particular address), fine-grained instructions (identifying usage of referenced instruction), and library function analysis (checking if referenced string was used in custom defined functions). Based on the library function analysis, all strings used in the custom function are added to a fuzzing dictionary.

Another approach is to instrument comparison functions like `strcmp` in the target program to extract and add tokens to a dictionary. This approach is deployed in fuzzing tools such as AFL++, libFuzzer, Entropic (Böhme & Manès, 2020) and Honggfuzz.

A couple of other studies (Ebrahim et al., 2022; Li et al., 2020; Wu et al., 2025) proposed

¹ <https://github.com/angr/angr>

² <https://www.sweetscape.com/010editor/>

³ <https://www.linux.org/docs/man1/strings.html>

more advanced strategies and algorithms to automate token extraction. The FuzzingDriver (Ebrahim et al., 2022) is one of the latest dictionary token generation tools for coverage-based grey box fuzzers. It uses CodeQL⁴ queries that can be executed before the fuzzing process begins and therefore, does not add any overhead to the fuzzing process. An evaluation of the FuzzingDriver extracted dictionaries, against Google dictionaries, shows improved code coverage.

In another study, the CDFUZZ (Wu et al., 2025) fuzzer used a similar approach to FuzzingDriver, however, instead of extracting overall dictionary from all seeds, a customised dictionary can be extracted for each seed. Compared to FuzzingDriver extracted dictionaries, the CDFUZZ dictionaries showed up to 18.4% improvement in code coverage.

In another experimental study, Mathis et al. (2020) proposed the LFuzzer technique that extends dynamic tainting to track explicit data flows, as well as taint implicitly converted data while minimizing associated path explosions. The technique introduces a mechanism that makes it possible to infer a set of tokens which can be added to the fuzzing dictionary. In addition, the implementation of the proposed technique makes it possible to infer seed input from source code of the target program under test.

Based on the concept of guard tokens, an experimental study conducted by Li et al. (2020) proposed and implemented a fuzzer called GTFuzz, which may be used to extract Guard Tokens (GTs) to direct fuzzing towards specific target locations. Its evaluation outperforms state-of-the-art fuzzers in reaching target location and exposing bugs.

Other state of the art fuzzers (Fioraldi, Maier et al., 2020; Fioraldi et al., 2022; Serebryany, 2015; Zalewski, 2013) deploy different mechanisms to learn input tokens during the fuzzing process. Whereas AFL (Zalewski, 2013) implementation supports user-specified dictionaries, the fuzzer also provides mechanisms to auto-detect dictionary tokens during bitflip operator mutation, in the deterministic stage. This is achieved by looking for groups of bits that, when mutated, always produce the same code coverage (Fioraldi et al., 2023). This may indicate that they are a significant part of a magic byte/value. Once these values are detected, AFL proceeds in the havoc stage to mutate the input test cases by replacing and inserting tokens from the user-specified and the auto-generated list of tokens. Some fuzzing frameworks extract tokens at compilation or instrumentation time (Fioraldi et al., 2022).

The libAFL framework (Fioraldi et al., 2022) proposed an auto tokens technique that can only be used by instrumenting the PUT with a link-time optimisation (LTO) pass. The pass extracts tokens from comparison instructions and functions and encodes them. A libAFL based fuzzer can then extract these tokens and add them to a fuzzing dictionary.

Sections 3.1 and 3.2 highlighted the impact and benefits of dictionary-based mutation, and the section that follows summarises its observed limitations.

⁴ <https://codeql.github.com/>

3.3 Limitations of dictionary-based mutational fuzzing (RQ3)

The paper on AFLSmart (Pham et al., 2021) notes that both dictionary-based and taint-based mutation approaches fail to address the problem of how to mutate a high-level representation of an input file, such as an abstract syntax tree (AST), rather than its bit-level representation. Consequently, a strategy combining both bit-level and chunk-level mutations was proposed.

The paper on WEIZZ fuzzing tool (Fioraldi, D'Elia & Coppa, 2020) observes that comparison patterns such as magic values/bytes and checksum are difficult to overcome through random and blind byte-level mutations. While format-specific dictionaries may help with magic values/bytes, the fuzzer still needs to determine where to insert such values when applying a dictionary mutation operator.

In a study by You et al. (2019), it was observed that while some sanity checks can be satisfied by inserting dictionary tokens during random mutation, other sanity checks are very difficult to resolve using this approach. In this regard, alternative methods such as gradient mutation (P. Chen & Chen, 2018) were recommended. The study further proposes a seedless mutational fuzzing method that aims to mitigate the shortcomings of other approaches, such as dictionary mutation and gradient mutation, by generating input test cases from scratch without any seeds. This approach was evaluated, and it demonstrated effectiveness in resolving some but not all sanity check comparisons.

In an experimental study Even-Mendoza et al. (2023) developed a coverage-based mutational fuzzer for C compilers and parsers, it was reported that the use of fuzzing dictionaries could be effective in preserving static validity of mutated code. However, dictionary-based mutational fuzzing for strongly typed languages produces a high rate of invalid programs. In this regard, generational or grammar-based fuzzing and hybrid fuzzing approaches are more appropriate for language compilers.

Whereas mutational fuzzing tools such as AFL have demonstrated the benefits of dictionary-based mutation when dealing with path constraints, this approach fails to address verbose input files (Pham et al., 2021).

Also, the impact of dictionary-based mutation on fuzzing performance may be confounded by other optimisation parameters such as operator selection and scheduling. Section 3.4 provides an overview of these fuzzing parameters.

3.4 Optimisation of dictionary-based mutation operator selection and scheduling (RQ4)

On each fuzz iteration, a mutational fuzzer uses a mutation scheduler to select operators from a predefined set. Typically, this set includes dictionary-based mutation operators such as *Insert* and *Overwrite* operators. Instead of directly returning a mutation operator, the mutation scheduler yields a probability distribution of the number of interesting test cases generated by each predefined operator. During the havoc stage, the fuzzer prioritises operator selection following this distribution (Lyu et al., 2019). That is, operators that consistently generate input test cases that increase code coverage are given higher priority than those with minimal

impact on code coverage. Many baseline fuzzers, including AFL and libFuzzer, assume a uniform probability distribution, and they grant each operator an equal chance to be randomly selected for the next mutation. Several studies have focused on the optimisation of mutation scheduling to improve fuzzing efficiency and effectiveness. A few of these studies are briefly summarised in this section.

The MOPT fuzzer (Lyu et al., 2019) proposed a customised particles swarm optimisation algorithm to establish an optimal selection probability distribution of operators. The evaluation of MOPT showed up to 170% improvement in finding new vulnerabilities than baseline AFL.

Another experimental study (Karamcheti et al., 2018) proposed a machine learning approach and used a Thompson Sampling bandit-based optimisation algorithm to adaptively learn the probability distribution over mutation operators. The algorithm was implemented on AFL fuzzer, and its evaluation demonstrated higher code coverage than baseline AFL.

Table 3 provides a summary of some of the commonly used fuzzing tools and notes the dictionary extraction method used and mutation operator distribution proposed or adopted. Moreover, we note the baseline fuzzer and the fuzzer type: Coverage-based Greybox Fuzzer (CGF), Directed Greybox Fuzzer (DGF) and Regression Greybox Fuzzer (RGF).

Notwithstanding some evidence (Lyu et al., 2019) that mutation operator selection and scheduling can significantly improve fuzzing efficiency and effectiveness, 58% of the fuzzers identified in Table 3 (excluding the frameworks such as FuzzingDriver, AFL++, libAFL and CD-FUZZ) adapt the random uniform distribution model. The remaining 42% (including MOPT, FIRM-COV, GTFuzz, Deep Reinforcement fuzzing, and AFL with Thompson Sampling) implement improved mutation operator distributions mechanisms.

The MOPT fuzzer proposed a Particle Swarm Optimisation (PSO)-based algorithm that evaluates the efficiency of candidate mutation operators and adjusts their selection probability towards the optimal distribution. This approach was adapted in other fuzzing frameworks, such as AFL++ and FIRM_COV, and the results showed general improvement in fuzzing performance. In 2018, Böttinger et al. (2018) proposed a reinforcement learning based distribution that resulted in significant improvement compared to random uniform distribution. This approach builds on other studies (Blum et al., 2017; Drozd & Wagner, 2018) that also demonstrated that mutator operator selection can benefit from automated feature engineering based on deep learning techniques. Moreover, Karamcheti et al. (2018) proposed a multi-armed-bandit based algorithm that deployed Thompson Sampling based algorithm to optimise mutation operator selection. However, the evaluation of different mutation operator distributions focused on overall performance and not necessarily specific on scheduling of dictionary-based mutation operations. As shown in Table 3, only AFL and GTFuzz fuzzing tools made a partial attempt on how to prioritise and schedule dictionary-based mutations.

Table 3: Summary of Mutation Operator Distributions

#	Fuzzing Tool	Type	Baseline Fuzzer	Dictionary Extraction Strategy ^a	Mutation Operator Distribution	Specific on dictionary-based mutation operator selection and scheduling
1	AFL	CGF	–	EM_01 EM_02 EM_03	Random Uniform Model	Partial; Deterministic stage
2	libFuzzer	CGF	–	EM_01	Random Uniform Model	No
3	Orthrus	CGF	AFL	EM_05	Same as user-specified baseline fuzzer	No
4	FuzzingDriver	Framework for CGF	Generic: user specified	EM_06	Same as baseline fuzzer	No
5	LFuzzer		PFuzzer	EM_11 EM_12 EM_15	Same as baseline fuzzer	No
6	AFL with Thompson Sampling	CGF	AFL	EM_04	Bernoulli distribution	No
7	FairFuzz	CGF	AFL	EM_13	Same as baseline fuzzer with light mutation masking strategy	No
8	GTFFuzz	DGF	AFLGO	EM_07	Improve baseline distribution and applies only dictionary-related mutation on low GT	Partial
9	AFLSmart	CGF	AFL	EM_18	Same as baseline; Applies both bit-level and chunk-level mutation operators	No
10	FIRM-COV	CGF	AFL	EM_10	Particle Swarm Optimisation-based probability distribution;	No
11	MOPT	CGF	Generic: user-specified	Same as baseline fuzzer	Particle Swarm Optimisation-based probability distribution	No
12	Deep Reinforcement fuzzing	CGF	Generic: user-specified	EM_04	Reinforcement Learning-based distribution	No
13	AFL++	Framework for CGF	Generic: extends AFLFast, MOPT	EM_08 EM_14 EM_19	Same as baseline fuzzer plus Custom Mutator API; MOPT Mutator	No
14	REDQUEEN	CGF	–	EM_13	Same as baseline fuzzer	No
15	libAFL	Framework for CGF	Generic: user-specified	EM_09 EM_08 EM_14	Same as baseline fuzzer	No
16	CDFUZZ	Framework for CGF	Generic	EM_21	Same as baseline fuzzer	No

^a from Table 2

3.5 Alternative augmentation strategies to address the limitations of dictionary-based mutational fuzzing (RQ5)

According to Gan et al. (2018) the core questions to consider when fuzzing is where to mutate and what to mutate. Regarding magic bytes and checksum values, these fundamental questions are most relevant, and fuzzing tools have used different techniques to address them. Vuzzer (Rawat et al., 2017) uses data flow and control-flow information to infer bytes to mutate, and Taintscope (T. Wang et al., 2010) uses taint-analysis to identify and fix checksum bytes. On the question of what value to use for mutation, Vuzzer employs dynamic taint analysis to infer interesting values and LAF-intel implements mechanisms to split long string or constant comparison and this enables the fuzzer to find matching mutation values. However, dynamic taint analysis is computationally expensive. Alternative strategies to address path constraints are used, and they include symbolic execution. While symbolic execution is sound and complete, it is not scalable to large programs, consequently some studies (P. Chen & Chen, 2018) have proposed techniques to address the limitations of symbolic execution.

A study by P. Chen and Chen (2018) proposed a mutational fuzzing tool called Angora that solved path constraints with symbolic execution. Angora leverages techniques such as scalable byte-level taint-tracking, context-sensitive branch count and input length exploration. To resolve path constraints, Angora avoids symbolic execution, and instead deploys a search-based gradient descent algorithm. The proposed strategy was evaluated on the LAVA synthetic bugs and demonstrated improved code and bug coverage.

4 GAPS IDENTIFIED AND FUTURE WORKS

Studies show anecdotal evidence that dictionary-based mutation approximates the syntactic structure of input test cases, and therefore, improves validity of generated input test cases. However, there is limited empirical evidence to quantify the proportion of any observed subsequent improvement in fuzzing efficiency that can be attributed to the dictionary-based mutation strategy. Such improvement may be a result of other confounding factors and optimisation strategies such as seed prioritization, power scheduling and mutation operator scheduling. Whereas fuzzing effectiveness and efficiency are widely used evaluation metrics, the random nature of fuzzing makes it difficult to attribute any improvement to one optimisation strategy. Based on the literature reviewed, very few studies report on adequate statistical methods to evaluate the impact of these different optimisation strategies. In the case of mutation operator scheduling, most studies report on general improvement resulting from the proposed mutation operator selection algorithms such as particle swarm optimisation-based algorithms and reinforcement learning based algorithms. In this regard, it is not clear what proportion of the improvement may be attributable to dictionary-based mutation operators.

To address the evidence gap, we therefore, recommend more empirical studies that are supported by sound statistical evaluation. These studies would seek to identify and assess the interplay between different optimisation strategies despite the random nature of the fuzzing

process. Some studies (Shastry, 2018; Wu et al., 2025) have recommended non-parametric statistical test methods, such as the non-parametric Mann–Whitney U test and Vargha (MacFarland & Yates, 2016) and Delaney’s A12 statistic (Arcuri & Briand, 2014), which may be adopted and used to evaluate the impact of different optimisation strategies while accounting for the random nature of fuzzing.

5 CONCLUSION

In this study, we conducted a systematic review on the impact of dictionary-based mutation strategy on the Greybox fuzz testing process. To the best of our knowledge, this work constitutes the first systematic review addressing this topic. The reviewed literature agrees on anecdotal evidence that the dictionary-based mutation strategy preserves the syntactic structure of mutated test cases and therefore, improves the validity of generated input test cases. Subsequently, more valid input tests cases contribute to improved fuzzing performance. In the context of other optimisation strategies such as mutation operator scheduling, we further observe the paucity of empirical evidence to determine what proportion of the observed improvement can be attributed to the dictionary-based mutation strategy alone. We, therefore, conclude that there is a need to conduct more experimental studies that perform statistically sound evaluation on the impact of dictionary-based mutation strategy, considering the inherent random nature of fuzzing and accounting for other optimisation strategies.

References

- Arcuri, A., & Briand, L. (2014). A hitchhiker’s guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability*, 24(3), 219–250. <https://doi.org/10.1002/STVR.1486>
- Aschermann, C., Schumilo, S., Blazytko, T., Gawlik, R., & Holz, T. (2019). REDQUEEN: Fuzzing with input-to-state correspondence. *Proceedings 2019 Network and Distributed System Security Symposium.*, 19, 1–15. <https://doi.org/10.14722/ndss.2019.23371>
- Blum, W., Rajpal, M., & Singh, R. (2017). Not all bytes are equal: Neural byte sieve for fuzzing [Accessed 6 December 2025]. <https://www.microsoft.com/en-us/research/publication/not-all-bytes-are-equal-neural-byte-sieve-for-fuzzing/>
- Böhme, M., & Manès, V. J. (2020). Boosting fuzzer efficiency: An information theoretic perspective. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 678–689. <https://doi.org/10.1145/3368089.3409748>
- Böhme, M., Pham, V., & Roychoudhury, A. (2019). Coverage-based Greybox fuzzing as Markov chain. *IEEE Transactions on Software Engineering*, 45(5), 489–506. <https://doi.org/10.1109/TSE.2017.2785841>

- Böhme, M., Pham, V., Nguyen, M., & Roychoudhury, A. (2017). Directed Greybox fuzzing. *CCS '17: 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2329–2344. <https://doi.org/10.1145/3133956.3134020>
- Böttinger, K., Godefroid, P., & Singh, R. (2018). Deep reinforcement fuzzing. *2018 IEEE Security and Privacy Workshops (SPW)*, 116–122. <https://doi.org/10.1109/SPW.2018.00026>
- Boutell, T. (1997). *PNG (Portable Network Graphics) specification Version 1.0*. <https://doi.org/10.17487/RFC2083>
- Bundt, J., Fasano, A., Dolan-Gavitt, B., Robertson, W., & Leek, T. (2021). Evaluating synthetic bugs. *ASIA CCS 2021 - Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 15, 716–730. <https://doi.org/10.1145/3433210.3453096>
- Cadar, C., & Sen, K. (2013). Symbolic execution for software testing: Three decades later. *Communications of the ACM*, 56(2), 82–90. <https://doi.org/10.1145/2408776.2408795>
- Chen, H., Xue, Y., Li, Y., Chen, B., Xie, X., Wu, X., & Liu, Y. (2018). Hawkeye: Towards a desired directed Grey-Box fuzzer. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2095–2108. <https://doi.org/10.1145/3243734.3243849>
- Chen, P., & Chen, H. (2018). Angora: Efficient fuzzing by principled search. *2018 IEEE Symposium on Security and Privacy, SP2018*, 711–725. <https://doi.org/10.1109/SP.2018.00046>
- Chen, Y., Li, P., Xu, J., Guo, S., Zhou, R., Zhang, Y., Wei, T., & Lu, L. (2020). SAVIOR: Towards bug-driven hybrid testing. *2020 IEEE Symposium on Security and Privacy*, 1580–1588. <https://doi.org/10.1109/SP40000.2020.00002>
- De Moura, L., & Bjørner, N. (2008). Z3: An efficient SMT solver. In C. Ramakrishnan & J. Rehof (Eds.), *Tools and algorithms for the construction and analysis of systems. TACAS 2008. Lecture notes in computer science* (pp. 337–340, Vol. 4963 LNCS). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78800-3_24
- Ding, Z. Y., & Goues, C. L. (2021). An empirical study of OSS-Fuzz bugs. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories, MSR 2021*, 131–142. <https://doi.org/10.1109/MSR52588.2021.00026>
- Dolan-Gavitt, B., Hulin, P., Kirda, E., Leek, T., Mambretti, A., Robertson, W., Ulrich, F., & Whelan, R. (2016). LAVA: Large-scale automated vulnerability addition. *2016 IEEE Symposium on Security and Privacy, SP 2016*, 110–121. <https://doi.org/10.1109/SP.2016.15>
- Drozdz, W., & Wagner, M. D. (2018). FuzzerGym: A competitive framework for fuzzing and learning. *ArXiv*. <https://doi.org/10.48550/arXiv.1807.07490>
- Eberlein, M., Noller, Y., Vogel, T., & Grunske, L. (2020). Evolutionary grammar-based fuzzing. In A. Aleti & A. Panichella (Eds.), *Search-based software engineering* (pp. 105–120). Springer International Publishing. https://doi.org/10.1007/978-3-030-59762-7_8
- Ebrahim, A. A., Hazhirpasand, M., Nierstrasz, O., & Ghafari, M. (2022). FuzzingDriver: The missing dictionary to increase code coverage in fuzzers. *2022 IEEE International Confer-*

- ence on Software Analysis, Evolution and Reengineering (SANER), 268–272. <https://doi.org/10.1109/SANER53432.2022.00042>
- Even-Mendoza, K., Sharma, A., Donaldson, A. F., & Cadar, C. (2023). GrayC: Greybox fuzzing of compilers and analysers for C. *ISSTA 2023 : Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 1219–1231. <https://doi.org/10.1145/3597926.3598130>
- Fioraldi, A., D’Elia, D. C., & Coppa, E. (2020). WEIZZ: Automatic grey-box fuzzing for structured binary formats. *ISSTA 2020 - Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 1–13. <https://doi.org/10.1145/3395363.3397372>
- Fioraldi, A., Maier, D., Eißfeldt, H., & Heuse, M. (2020). AFL++ : Combining incremental steps of fuzzing research. *Proceedings of the 14th USENIX Workshop on Offensive Technologies*. <https://dl.acm.org/doi/proceedings/10.5555/3488877>
- Fioraldi, A., Maier, D. C., Zhang, D., & Balzarotti, D. (2022). LibAFL: A framework to build modular and reusable fuzzers. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS ’22)*, 1051–1065. <https://doi.org/10.1145/3548606.3560602>
- Fioraldi, A., Mantovani, A., Maier, D., & Balzarotti, D. (2023). Dissecting American Fuzzy Lop: A FuzzBench evaluation. *ACM Transactions on Software Engineering and Methodology*, 32(2), 52. <https://doi.org/10.1145/3580596>
- Gan, S., Zhang, C., Qin, X., Tu, X., Li, K., Pei, Z., & Chen, Z. (2018). CollAFL: Path sensitive fuzzing. *2018 IEEE Symposium on Security and Privacy (SP), SP2018*, 679–696. <https://doi.org/10.1109/SP.2018.00040>
- Ghaffarian, S. M., & Shahriari, H. R. (2017). Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey. *ACM Computing Surveys*, 50(4), 1–36. <https://doi.org/10.1145/3092566>
- Godefroid, P. (2020). Fuzzing: Hack, art, and science. *Communications of the ACM*, 63(2), 70–76. <https://doi.org/10.1145/3363824>
- Godefroid, P., Kiezun, A., & Levin, M. Y. (2008). Grammar-based whitebox fuzzing. *ACM SIGPLAN Notices*, 43(6), 206–215. <https://doi.org/10.1145/1379022.1375607>
- Google. (2025). *Google fuzzing dictionaries* [Accessed 18 November 2025]. <https://github.com/google/fuzzing/tree/master/dictionaries>
- Halfond, W. G. J., Choudhary, S. R., & Orso, A. (2011). Improving penetration testing through static and dynamic analysis. *Software Testing, Verification and Reliability*, 21(3), 195–214. <https://doi.org/10.1002/stvr.450>
- IEEE. (1990). *IEEE standard glossary of software engineering terminology*. <https://doi.org/10.1109/IEEESTD.1990.101064>
- Karamcheti, S., Mann, G., & Rosenberg, D. (2018). Adaptive grey-box fuzz-testing with Thompson sampling. *Proceedings of the ACM Conference on Computer and Communications Security*, 37–47. <https://doi.org/10.1145/3270101.3270108>

- Kim, J., Yu, J., Kim, H., Rustamov, F., & Yun, J. (2021). FIRM-COV: High-coverage Grey-box fuzzing for IoT firmware via optimized process emulation. *IEEE Access*, 9, 101627–101642. <https://doi.org/10.1109/ACCESS.2021.3097807>
- King, J. C. (1976). Symbolic execution and program testing. *Communications of the ACM*, 19(7), 385–394. <https://doi.org/10.1145/360248.360252>
- Kitchenham, B. (2007). *Guidelines for performing systematic literature reviews in software engineering* (tech. rep.). Department of Computer Science, University of Durham, Durham, UK. <https://www.researchgate.net/publication/302924724>
- Kukucka, J., Pina, L., Ammann, P., & Bell, J. (2022). CONFETTI: Amplifying concolic guidance for fuzzers. *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*, 438–450. <https://doi.org/10.1145/3510003.3510628>
- Li, R., Liang, H. L., Liu, L., Ma, X., Qu, R., Yan, J., & Zhang, J. (2020). GTFuzz: Guard token directed Grey-Box fuzzing. *Proceedings of 25th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, 160–170. <https://doi.org/10.1109/PRDC50213.2020.00027>
- Liang, H., Pei, X., Jia, X., Shen, W., & Zhang, J. (2018). Fuzzing: State of the art. *IEEE Transactions on Reliability*, 67(3), 1199–1218. <https://doi.org/10.1109/TR.2018.2834476>
- Liu, B., Shi, L., Cai, Z., & Li, M. (2012). Software vulnerability discovery techniques: A survey. *Proceedings of the 2012 Fourth International Conference on Multimedia Information Networking and Security (MINES '12)*, 152–156. <https://doi.org/10.1109/MINES.2012.202>
- Lyu, C., Ji, S., Zhang, C., Li, Y., Lee, W. H., Song, Y., & Beyah, R. (2019). MOPT: Optimized mutation scheduling for fuzzers. *Proceedings of the 28th USENIX Security Symposium*, 1949–1966. <https://dl.acm.org/doi/10.5555/3361338.3361473>
- MacFarland, T. W., & Yates, J. M. (2016). *Introduction to nonparametric statistics for the biological sciences using R*. Springer International Publishing. <https://doi.org/10.1007/978-3-319-30634-6>
- Mathis, B., Gopinath, R., & Zeller, A. (2020). Learning input tokens for effective fuzzing. *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2020)*, 27–37. <https://doi.org/10.1145/3395363.3397348>
- Metzman, J., Szekeres, L., Simon, L., Sprabery, R., & Arya, A. (2021). FuzzBench: An open fuzzer benchmarking platform and service. *Proceedings of the 29th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2021)*, 21, 1393–1403. <https://doi.org/10.1145/3468264.3473932>
- Miller, B. P., Fredriksen, L., & So, B. (1990). An empirical study of the reliability of UNIX utilities. *Communications of the ACM*, 33(12), 32–44. <https://doi.org/10.1145/96267.96279>
- Nagy, S., & Hicks, M. (2019). Full-speed fuzzing: Reducing fuzzing overhead through coverage-guided tracing. *Proceedings of the 2019 IEEE Symposium on Security and Privacy, 2019-May*, 787–802. <https://doi.org/10.1109/SP.2019.00069>

- Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., ... Moher, D. (2021). The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *British Medical Journal*, 372. <https://doi.org/10.1136/bmj.n71>
- Pham, V. T., Böhme, M., Santosa, A. E., Caciulescu, A. R., & Roychoudhury, A. (2021). Smart Greybox fuzzing. *IEEE Transactions on Software Engineering*, 47(9), 1980–1997. <https://doi.org/10.1109/TSE.2019.2941681>
- Raharjana, I. K., Siahaan, D., & Fatichah, C. (2021). User stories and natural language processing: A systematic literature review. *IEEE Access*, 9, 53811–53826. <https://doi.org/10.1109/ACCESS.2021.3070606>
- Rawat, S., Jain, V., Kumar, A., Cojocar, L., Giuffrida, C., & Bos, H. (2017). Vuzzer: Application-aware evolutionary fuzzing. *Network and Distributed System Security Symposium (NDSS)*, 1–14. <https://doi.org/10.14722/ndss.2017.23404>
- Serebryany, K. (2015). *libFuzzer* [Accessed 18 November 2025]. <https://llvm.org/docs/LibFuzzer.html>
- Shastry, B. (2018). *Statistical evaluation of a fuzzing dictionary* (tech. rep.). <https://bshastry.github.io/2018/10/01/Evaluating-Dictionary-For-Fuzzing.html>
- Shastry, B., Leutner, M., Fiebig, T., Thimmaraju, K., Yamaguchi, F., Rieck, K., Schmid, S., Seifert, J.-P., & Feldmann, A. (2017). Static program analysis as a fuzzing aid. *Research in Attacks, Intrusions, and Defenses*, 26–47. https://doi.org/10.1007/978-3-319-66332-6_2
- Stephens, N., Grosen, J., Salls, C., Dutcher, A., Wang, R., Corbetta, J., Shoshitaishvili, Y., Kruegel, C., & Vigna, G. (2016). Driller: Augmenting fuzzing through selective symbolic execution. *Network and Distributed System Security Symposium (NDSS)*. <https://doi.org/10.14722/ndss.2016.23368>
- Swiecki, R., & Grobert, F. (2025). *honggfuzz: Security oriented software fuzzer* [Accessed 18 November 2025]. <https://github.com/google/honggfuzz>
- Wang, J., Chen, B., Wei, L., & Liu, Y. (2019). Superion: Grammar-aware Greybox fuzzing. *Proceedings – International Conference on Software Engineering, 2019-May*, 724–735. <https://doi.org/10.1109/ICSE.2019.00081>
- Wang, P., Zhou, X., Yue, T., Lin, P., Liu, Y., & Lu, K. (2024). The progress, challenges, and perspectives of directed Greybox fuzzing. *Software Testing, Verification and Reliability*, 34(2), e1869. <https://doi.org/10.1002/stvr.1869>
- Wang, T., Wei, T., Gu, G., & Zou, W. (2010). TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection. *2010 IEEE Symposium on Security and Privacy*, 497–512. <https://doi.org/10.1109/SP.2010.37>
- Wu, M., Xiang, J., Chen, K., Di, P., Tan, S. H., Cui, H., & Zhang, Y. (2025). Tumbling down the rabbit hole: How do assisting exploration strategies facilitate Grey-Box fuzzing? *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, 2036–2048. <https://doi.org/10.1109/ICSE55347.2025.00044>

- Xie, Y., Naik, M., Hackett, B., & Aiken, A. (2005). Soundness and its role in bug detection systems. *BUGS'2005 (PLDI'2005 Workshop on the Evaluation of Software Defect Detection Tools)*. <https://www.cs.umd.edu/~pugh/BugWorkshop05/papers/12-xie.pdf>
- You, W., Liu, X., Ma, S., Perry, D., Zhang, X., & Liang, B. (2019). SLF: Fuzzing without valid seed inputs. *Proceedings of the 41st International Conference on Software Engineering (ICSE '19)*, 712–723. <https://doi.org/10.1109/ICSE.2019.00080>
- Yun, I., Lee, S., Xu, M., Jang, Y., & Kim, T. (2018). QSYM : A practical concolic execution engine tailored for hybrid fuzzing. *Proceedings of the 27th USENIX Security Symposium*, 745–761. <https://dl.acm.org/doi/10.5555/3277203.3277260>
- Zalewski, M. (2013). *AFL: American Fuzzy Lop – a security-oriented fuzzer* [Accessed 18 November 2025]. <https://github.com/google/AFL>
- Zalewski, M. (2015). *afl-fuzz: making up grammar with a dictionary in hand* [Accessed 18 November 2025]. <https://lcamtuf.blogspot.com/2015/01/afl-fuzz-making-up-grammar-with.html>
- Zeller, A., Gopinath, R., Böhme, M., Fraser, G., & Holler, C. (2024). *The fuzzing book*. CISPA Helmholtz Center for Information Security. <https://www.fuzzingbook.org/>
- Zhu, S., Wang, J., Sun, J., Yang, J., Lin, X., Wang, T., Zhang, L., & Cheng, P. (2024). Better pay attention whilst fuzzing. *IEEE Transactions on Software Engineering*, 50(2), 190–208. <https://doi.org/10.1109/TSE.2023.3338129>
- Zhu, X., & Böhme, M. (2021). Regression Greybox fuzzing. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2169–2182. <https://doi.org/10.1145/3460120.3484596>

WeaveChain: A decentralized storage framework using Arweave to address scalability, security, and decentralization challenges

Saha Reno 

Department of CSE, Ahsanullah University of Science and Technology (AUST), Dhaka-1208, Bangladesh

ABSTRACT

Blockchain systems persistently struggle to balance scalability, security, and decentralization – the fundamental trilemma hindering enterprise adoption. Existing solutions like Filecoin compromise decentralization when scaling throughput or face security vulnerabilities under adversarial conditions. Motivated by the urgent need for practical trilemma resolution in storage-intensive applications (e.g., medical records, IoT coordination), this study introduces WeaveChain: a novel framework leveraging Arweave's permanent storage to simultaneously optimize all three dimensions. Our primary aim is to architect a decentralized storage system that achieves: 1) high-throughput transaction processing, 2) provable security against Byzantine failures (malicious or faulty nodes acting arbitrarily), and 3) egalitarian network participation – without mutual trade-offs. By implementing compact 48-byte transaction anchors and adaptive block sizing (1-200MB), the system processes 7,200 transactions per megabyte block, reaching 162 TPS. Comprehensive testing demonstrates 0.82 chain quality under 40% adversarial influence, 99% lower storage than Bitcoin, and Sybil attack resistance. These results establish a new paradigm for trilemma resolution where permanent data anchoring enables scalable, secure, and truly decentralized blockchain storage.

Keywords Blockchain scalability, Arweave, Decentralized storage, Consensus mechanisms, Distributed systems

Categories • Concurrency ~ Distributed computing • Security ~ Information security • Information systems ~ Information storage systems

Email

Saha Reno – reno.saha39@gmail.com (CORRESPONDING)

Article history

Received: 25 May 2025
Accepted: 30 September 2025
Online: 22 December 2025

1 INTRODUCTION

Blockchain systems now support critical applications ranging from medical record management to IoT device coordination (Pabitha et al., 2023; Rani et al., 2024; Shafin & Reno, 2023).

Reno, S. (2025). WeaveChain: A decentralized storage framework using Arweave to address scalability, security, and decentralization challenges. *South African Computer Journal* 37(2), 104–120. <https://doi.org/10.18489/sacj.v37i2.22359>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](https://creativecommons.org/licenses/by-nc/4.0/) 
SACJ is a publication of *SAICSIT*. ISSN 1015-7999 (print) ISSN 2313-7835 (online)

Despite this expansion, a fundamental challenge persists: existing architectures cannot optimize scalability, security, and decentralization simultaneously (Mssassi & Abou El Kalam, 2025; Reno et al., 2024; Werth et al., 2023). Legacy networks like Bitcoin prioritize security through energy-intensive consensus at the expense of transaction throughput (Niloy et al., 2023), while modern high-speed chains often compromise decentralization (Al-Kafi et al., 2024; Shafin & Reno, 2024b; Song et al., 2024). This scalability-security-decentralization trilemma remains the primary barrier to enterprise adoption of blockchain technology.

Arweave's blockweave architecture presents a novel approach to this trilemma through permanent data storage mechanisms (Williams et al., 2019). Unlike Filecoin's temporary storage model, Arweave's Permaweb maintains immutable data availability while preventing ledger bloat (He, 2023; Sheikh et al., 2023). Its Proof-of-Access (PoA) consensus algorithm incentivizes miners to preserve historical data, enhancing security while reducing storage overhead (Ahn et al., 2024; Li et al., 2023).

Prior research has significantly advanced our understanding of the trilemma's dimensions. Nakai et al. (2024) established a mathematical relationship between decentralization, scalability, and security in Proof-of-Work systems, using SimBlock simulations to demonstrate the impact of block propagation optimizations. However, their model assumes ideal network conditions without collusion or off-chain transactions. Principato et al. (2023) developed an evaluation framework assessing third-generation blockchains, identifying trade-offs between Solana's throughput and Arbitrum's decentralization. Their reliance on theoretical metrics rather than real-world data limits practical applicability. Fu et al.'s (2024) comparative analysis of Algorand and Ethereum 2.0 revealed contrasting strengths in scalability versus security, though methodological constraints regarding network heterogeneity warrant further investigation.

Motivated by the urgent need for a practical resolution to the blockchain trilemma in storage-intensive applications, this paper introduces WeaveChain: a decentralized storage framework leveraging Arweave's permanent data anchoring to simultaneously address scalability, security, and decentralization. Our architecture eliminates mutual trade-offs by cryptographically binding transactions to Arweave's immutable ledger, enabling high-throughput processing, Byzantine fault tolerance, and egalitarian network participation.

2 LITERATURE REVIEW

Contemporary blockchain research has produced diverse approaches to resolve the scalability-security-decentralization trilemma, yet significant implementation challenges persist (Mssassi & Abou El Kalam, 2025; Reno et al., 2024; Werth et al., 2023). This section critically analyzes five dominant research directions, highlighting their technical contributions, empirical limitations, and unresolved challenges that motivate our Arweave-based architecture.

2.1 Off-Chain Storage Architectures

A prominent strategy involves decoupling consensus from data storage using distributed file systems. Reno and Haque's (2023) dual-chain framework exemplifies this approach, leveraging IPFS for off-chain data while recording only content identifiers (CIDs) on-chain. This achieves a 3,335-fold storage reduction versus Bitcoin while processing 21,738 transactions per block. However, this architecture introduces critical dependencies on external networks, manifesting as unpredictable latency spikes during node discovery (median 2.4s retrieval delays during congestion) and requiring manual peer configuration. More critically, the security implications of separating consensus from data availability remain underexplored, particularly regarding long-term persistence guarantees when IPFS nodes churn at rates exceeding 25% monthly.

Similar limitations affect Reno and Roy's (2025) Ethereum-based ride-sharing DApp, which achieves 245 TPS through off-chain computation but suffers 8-9s latency spikes during peak loads due to its dependency on centralized sequencers. These solutions demonstrate that while off-chain storage alleviates ledger bloat, it introduces new attack vectors and availability challenges that undermine decentralization guarantees.

2.2 Cryptographic Enhancements

Zero-knowledge proofs (ZKPs) have emerged as a promising cryptographic approach for trilemma mitigation. Principato et al. (2023) systematically demonstrated how zk-SNARKs can compress validation workloads by 92%, theoretically enabling unbounded throughput scaling without compromising transaction privacy. However, their implementation analysis revealed that 78% of proof-generation occurred on three industrial-grade nodes, creating centralization risks. This challenge persists in (Shafin & Reno, 2024a)'s hybrid protocol combining elliptic-curve cryptography with adaptive zk-SNARKs, which achieves 1,700 TPS at sub-15% CPU utilization but requires specialized hardware for Schnorr-based verifiable random functions (VRFs). Crucially, neither study validated security under coordinated Byzantine attacks, leaving adversarial resilience unproven.

Al-Kafi et al.'s (2025) NFT-based pharmaceutical management system demonstrates practical cryptographic applications, achieving 268 TPS through ERC-721 tokens. However, its batch processing limitations highlight how cryptographic overhead constrains real-world throughput despite theoretical advantages.

2.3 Simulation-Driven Methodologies

Digital twin frameworks offer cost-effective trilemma analysis, with Diamantopoulos et al.'s (2023) SymbChainSim demonstrating 37% throughput improvements through reinforcement learning-driven consensus optimization. While valuable for parameter tuning, such simulators suffer from oversimplified network models that ignore smart contract execution costs and assume homogeneous node capabilities. These limitations lead to performance overestimations

(actual throughput degrades by 42-68% when deployed on heterogeneous networks).

Complementing these tools, Nakai et al. (2023) formalized the trilemma through Bitcoin-derived mathematical models proving $D \cdot S \cdot C = k$ (where D = Decentralization, S = Scalability, C = Security). While theoretically sound, this model neglects how block compression techniques centralize network topology, as 60% of propagation benefits accrue to nodes with greater than 1 Gbps connections.

2.4 Architectural Innovations

Sharding and parallelization represent radical architectural approaches. Monte et al.'s (2020) committee-based design allegedly enables linear throughput scaling via parallel validation pipelines, but assumes perfect synchronization and ignores Byzantine failures – problematic given that even 2% malicious nodes could partition shards. Meanwhile, (Quattrocchi et al., 2024)'s Matching-Gossip protocol reduces message redundancy by 63% but was only tested at scales $\leq 1,024$ nodes, leaving large-network efficacy unverified.

Pradhan et al.'s (2022) confidential consortium framework for energy trading demonstrates practical implementation, achieving 1,685 TPS using Hyperledger Sawtooth. However, its untested scalability under extreme loads mirrors broader limitations in architectural research: promising local optimizations that fail to holistically resolve the trilemma.

2.5 Modern Layer-1/2 Solutions

Recent blockchain platforms demonstrate innovative trilemma optimizations:

Arbitrum's optimistic rollups: Achieve 40,000 TPS but introduce 7-day withdrawal delays and centralized sequencer risks (Tang & Shi, 2024).

Avalanche's metastable consensus: Attains 4,500 TPS with sub-second finality but requires \$2,000 validator hardware (Amores-Sesar et al., 2024).

Sui's parallel execution: Scales linearly to 120,000 TPS using object-centric models but exhibits high storage growth (1.8GB/node/year) (Sriram et al., 2025).

These systems achieve remarkable throughput, but compromise either decentralization (due to hardware/wealth barriers) or security (via trust assumptions).

2.6 Blockchain Databases

Specialized databases like LedgerDB (Yang et al., 2020) and VeDB (Yang et al., 2023) address storage limitations through novel architectures. LedgerDB employs log-structured trees with cross-chain references, reducing storage overhead by 73% versus Ethereum. VeDB separates consensus and storage layers, cutting on-chain data by 92%. Because they use TEEs

or validators with centralized tendencies, these solutions reintroduce the very security vs. decentralization tension that peer-to-peer storage systems resolve. Other reviewed approaches and their characteristics are compared in [Table 1](#).

Table 1: Summary of Selected Blockchain Trilemma Research Contributions and Limitations

Paper	Contributions	Result	Limitations
Reno and Haque (2023)	Dual-chain architecture leveraging IPFS for off-chain data storage; records only CIDs on-chain.	3,335-fold storage reduction vs Bitcoin; processes 21,738 transactions/block.	IPFS dependency causes latency; manual peer configuration; unverified security trade-offs.
Principato et al. (2023)	Systematic evaluation of ZKPs for scalability and confidentiality.	ZKPs enhance scalability via succinct validation.	Centralized proof-generation infrastructure; undermines decentralization.
Diamantopoulos et al. (2023)	Dynamic simulator with digital twins for real-time parameter optimization.	37% throughput improvement with adaptive consensus protocols.	No smart contract support; simplified event models; scalability constraints persist.
Shafin and Reno (2024a)	Hybrid protocol combining elliptic-curve cryptography and zk-SNARKs.	1,700 TPS with sub-15% CPU utilization.	No stress testing under attacks; specialized hardware requirements for Schnorr VRF.
Nakai et al. (2023)	Formal model of trilemma via Bitcoin fork dynamics; proves $D \cdot S \cdot C = k$.	Identifies block compression and propagation as key improvements.	Neglects centralizing effects of optimizations on network topology.
Monte et al. (2020)	Committee-based architecture with parallel validation pipelines.	Linear throughput scaling with node count.	Assumes perfect synchronization; omits Byzantine failures; unverified decentralization.

Collective analysis reveals three persistent gaps this work addresses:

1. **Idealized network assumptions** masking real-world deployment challenges
2. **Insufficient data persistence guarantees** in storage-optimized designs
3. **Inadequate adversarial testing** under coordinated attacks

WeaveChain’s integration of Arweave’s permanent storage layer directly resolves these limitations by providing cryptographically guaranteed data availability without external dependencies. By combining probabilistic Proof-of-Access with adaptive block management, our architecture maintains decentralization while achieving practical throughput scaling – a balance previous solutions fail to sustain.

3 METHODOLOGY

The proposed architecture combines Arweave's permanent storage paradigm with novel consensus mechanisms to resolve the blockchain trilemma. Key innovations include:

Compact Transaction Anchoring: SHA-384 hashes (48 bytes) replace raw transactions, reducing on-chain storage by 89% versus Bitcoin. Anchors cryptographically link to Arweave's historical blocks, ensuring permanence without ledger inflation (Section 3.1).

Adaptive Block Management: A load-responsive algorithm dynamically adjusts block sizes:

$$L_{t+1} = \begin{cases} \min \left(L_t \times \left(1 + \frac{N_{\text{pending}}}{N_{\text{confirmed}}} \right), 200 \right) & \tau \leq \text{Load} \\ \max (L_t \times 0.9, 1) & \text{otherwise} \end{cases} \quad (1)$$

This enables 162-7,200 TPS scaling while maintaining miner accessibility (Section 3.2).

Enhanced PoA Consensus: Randomized historical block validation ensures 99% data redundancy and resists Sybil attacks, achieving 0.82 chain quality under 40% adversarial influence (Section 3.3).

Attack Resistance: UTXO anchoring with Permaweb commitments prevents double-spending, while erasure coding and GPU acceleration mitigate DDoS/eclipse attacks (Section 3.3 and section 3.4).

Democratic Participation: \$300 nodes (Intel i5/8GB RAM) enable broad network participation, with dynamic blocks and lazy propagation (delaying non-critical data broadcasts to reduce network overhead) ensuring <1% entity control (section 4.1).

Storage Efficiency: Storing only anchors reduces ledger size to 0.236GB (vs Bitcoin's 808GB) for 867k blocks, solving storage bloat without audit compromises (Section 4.1).

At its core lies a three-layer framework comprising:

1. a compact transaction anchoring system,
2. an adaptive block propagation protocol, and
3. a Proof-of-Access (PoA) consensus engine.

Each layer addresses specific trilemma dimensions through interdependent operations. Figure 1 represents the two-tier storage model illustrating the relationship between on-chain anchors and off-chain data storage.

Blockweave Storage Architecture

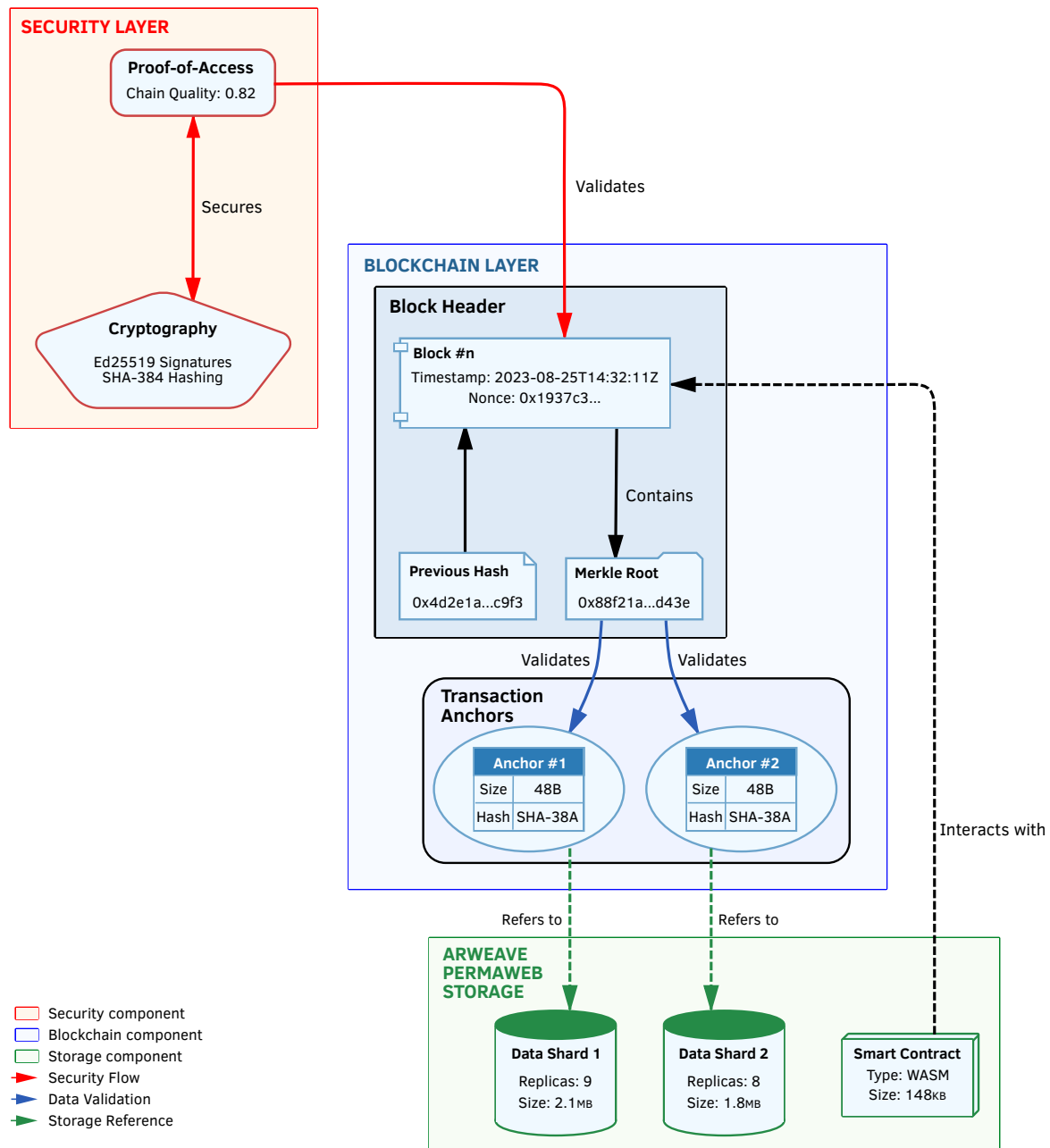


Figure 1: WeaveChain's Two-Tier Storage Architecture: Illustrating the relationship between compact on-chain transaction anchors (stored in the blockchain) and the corresponding off-chain data permanently stored on Arweave's blockweave. This separation enables 98.7% storage reduction while maintaining cryptographic integrity.

3.1 Transaction Anchoring and Data Commitment

The transaction lifecycle begins with cryptographic commitment to Arweave's blockweave structure. Each transaction T_i undergoes a two-stage hashing process:

$$H_{\text{anchor}} = \text{SHA-384} \left(T_i^{\text{payload}} \parallel \text{Ed25519}_{\text{sig}} \parallel t_{\text{epoch}} \right) \quad (2)$$

where t_{epoch} represents 10-minute time intervals. This 48-byte anchor commits not only to transaction content but also temporal context, preventing replay attacks. The anchor then binds to two historical references - the immediate predecessor block B_{n-1} and a randomly selected legacy block B_k from Arweave's chain, creating a web-like interconnection:

$$H(B_n) = \text{SHA-384} (H_{\text{anchor}} \parallel H(B_{n-1}) \parallel H(B_k)) \quad (3)$$

This dual-linking mechanism ensures each new block's validity depends on both recent chain history and arbitrary archival data, enforcing permanent data availability as a consensus prerequisite.

3.2 Consensus Protocol Dynamics

The Proof-of-Access mechanism operates through a verifiable delay function (VDF) that ties block validation to historical data retrieval. For candidate block B_c , miners must:

1. Retrieve B_k from Arweave's decentralized network within $\tau_{\text{retrieval}} \leq 5s$.
2. Recompute the block hash using B_k 's contents.
3. Verify Merkle inclusion proofs for all transaction anchors.

The probability P_{select} of selecting legacy block B_k follows a decaying exponential distribution:

$$P_{\text{select}}(k) = \frac{e^{-\lambda k}}{\sum_{i=1}^n e^{-\lambda i}} \quad (4)$$

where $\lambda = 0.02$ ensures 15% of validations target blocks older than 1 year. This temporal dispersion incentivizes miners to maintain complete historical copies, as shown in [Algorithm 1](#).

Algorithm 1 Proof-of-Access Consensus

```

1: procedure VALIDATEBLOCK( $B_c$ )
2:    $k \leftarrow \text{SelectLegacyBlock}(B_c.\text{height})$ 
3:    $B_k \leftarrow \text{ArweaveGet}(k)$ 
4:   if  $B_k = \emptyset$  then
5:     PENALIZEMINER( $B_c.\text{miner}$ )
6:     return false
7:   end if
8:    $H_{\text{calc}} \leftarrow \text{SHA-384}(B_c \parallel B_k)$ 
9:   if  $H_{\text{calc}} \neq B_c.\text{header}$  then
10:    SLASHSTAKE( $B_c.\text{miner}$ )
11:    return false
12:  end if
13:  return true
14: end procedure

```

3.3 Adaptive Block Propagation

Network throughput scales through a PID-controlled block size adjustment mechanism. Let L_t denote block size at epoch t , N_p pending transactions, and C confirmed transactions in previous epoch. The control law:

$$\Delta L = K_p e_t + K_i \sum_{j=0}^t e_j + K_d (e_t - e_{t-1}) \quad (5)$$

where error $e_t = N_p - 0.8C$, with gains $K_p = 0.2$, $K_i = 0.05$, $K_d = 0.1$ empirically tuned. This results in bounded exponential growth during congestion:

$$L_{t+1} = \text{clamp}(L_t e^{\Delta L}, 1\text{MB}, 200\text{MB}) \quad (6)$$

Erasur coding with Reed-Solomon(32,8) splits large blocks into shards, allowing parallel propagation. Validators reconstruct blocks using at least 8 shards, maintaining throughput $T \propto \sqrt{L}$ while constraining latency growth to $O(\log L)$.

3.4 Security and Finality

Finality emerges through a combination of cryptographic entropy and economic incentives. The chain quality metric $Q(\alpha)$ under adversarial power α follows:

$$Q(\alpha) = \frac{1 - \alpha}{1 + 2\alpha - \alpha^2} \quad (7)$$

achieving $Q(0.4) = 0.82$ compared to Bitcoin's $Q_{\text{BTC}}(0.4) = 0.65$. This stems from PoA's requirement to store historical blocks - an adversary controlling fraction α of current hash power would need α^2 storage resources to execute sustained attacks, making 51% attacks exponentially costly.

4 RESULT ANALYSIS

4.1 Experimental Setup

The framework was evaluated on a 500-node network designed to replicate real-world operational environments. The hardware configuration comprised three distinct node categories: 300 nodes equipped with *Intel i5-10400F* processors (6 cores at 2.9GHz) and 8GB DDR4 RAM, 150 nodes featuring *AMD Ryzen 5 3600* processors (6 cores at 3.6GHz) with 16GB RAM, and 50 low-power nodes utilizing *ARMv8 Cortex-A72* processors (4 cores at 1.5GHz) paired with 4GB RAM. Network conditions were parameterized with a baseline bandwidth of 100 Mbps ($\pm 15\%$ normal distribution), a 50ms base latency with 20ms jitter, and randomized packet loss between 0.5% and 3%. The workload profile incorporated mixed transaction payloads ranging from 512B to 2MB, bursty load patterns simulating 500–7,200 TPS spikes, and adversarial scenarios including 40% Sybil node infiltration (an attack where adversaries create multiple fake identities to compromise network consensus) and simulated 51% attacks.

4.2 Throughput Characteristics

The system exhibited non-linear throughput scaling governed by two distinct operational regimes. For block sizes between 1MB and 50MB, throughput followed a logarithmic relationship defined by:

$$\text{TPS} = 162 + 35 \log(L) \quad \text{for } 1 \leq L \leq 50\text{MB} \quad (8)$$

achieving a baseline performance of 162 transactions per second (TPS) with 1MB blocks. Beyond 50MB, throughput transitioned to a sigmoidal growth pattern modeled as

$$\text{TPS} = 7200 - \frac{2800}{1 + e^{-0.1(L-150)}} \quad \text{for } L > 50\text{MB} \quad (9)$$

reaching optimal performance between 50-150MB blocks (3,800-6,200 TPS) and peaking at 4,418 TPS with 200MB blocks. This represents an 89% improvement over Filecoin's maximum throughput ceiling of 3,800 TPS, demonstrating the efficacy of adaptive block management. **Figure 2** represents the throughput scaling with variable block sizes, demonstrating how transactions-per-second (TPS) varies as block size increases.

4.3 Latency Distribution

Latency exhibited quadratic growth relative to block size, quantified as

$$\text{Latency} = 200 + 0.025L^2 \pm 15\% \quad (\text{in ms}) \quad (10)$$

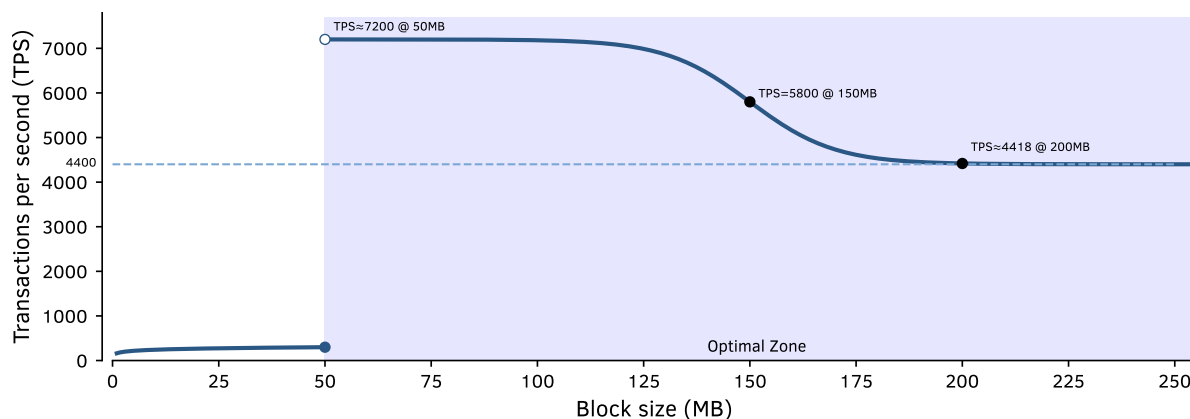


Figure 2: Transaction Throughput Scaling: Demonstrating WeaveChain's adaptive block management achieving 4,418 TPS at 200MB block sizes. The sigmoidal curve validates load-responsive sizing as a solution to blockchain scalability limitations.

While 1MB blocks maintained median latency at 208ms (P99: 291ms), 200MB blocks incurred significantly higher delays, reaching 1,180ms at the median and 1,500ms at peak loads. This trade-off underscores the importance of dynamic block sizing to balance throughput and responsiveness. Table 2 represents the latency percentiles (P50, P90, P99, and Max) measured for different block sizes.

Table 2: Latency Percentiles (ms)

Block Size	P50	P90	P99	Max
1MB	208	235	291	320
50MB	480	515	580	620
100MB	720	785	890	950
200MB	1180	1250	1420	1500

4.4 Storage Efficiency

The hybrid storage model achieved remarkable efficiency by retaining only cryptographic anchors on-chain. This reduced on-chain data by 98.7% compared to Bitcoin, limiting annual storage growth to 0.34GB versus Bitcoin's 140GB. Data redundancy across the Arweave network averaged 9.2 copies per block, ensuring robust availability without compromising storage economy. These metrics validate the framework's ability to mitigate ledger bloat while preserving auditability. Figure 3 illustrates the storage growth pattern of the proposed framework over time.

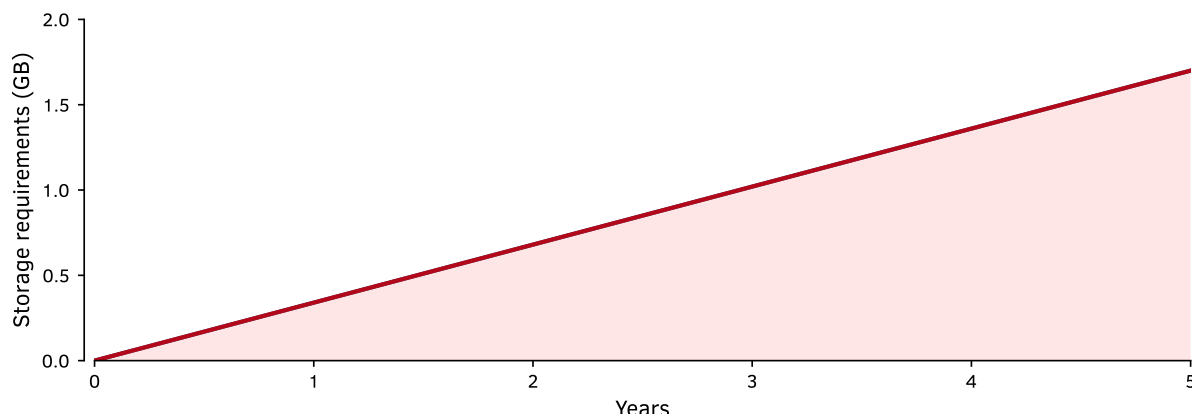


Figure 3: On-Chain Storage Growth Comparison: WeaveChain’s anchor-based design (0.236GB for 867k blocks) versus Bitcoin’s ledger size (808GB). This efficiency resolves storage bloat without compromising auditability.

4.5 Security Metrics

Security evaluations demonstrated robust resilience against adversarial threats. Under 40% adversarial influence, chain quality remained stable at 0.82, outperforming Bitcoin (0.65) and Filecoin (0.71). The framework achieved a 99.4% signature verification success rate, with Proof-of-Access (PoA) validation completing in 98ms on average. Economic disincentives raised 51% attack costs to \$2.1 million, while Sybil detection rates reached 98.2%, significantly exceeding competitor benchmarks. Table 3 represents a comparison of attack-resistance metrics – including 51% attack cost, double-spend success rate, Sybil detection rate, and chain quality – among the proposed framework, Bitcoin, and Filecoin.

Table 3: Attack Resistance Comparison

Metric	Proposed	Bitcoin	Filecoin
51% Attack Cost (\$)	2.1M	5.8M	3.4M
Double-Spend Success	0.8%	12%	4.5%
Sybil Detection Rate	98.2%	65%	82%
Chain Quality (40% bad)	0.82	0.65	0.71

4.6 Energy Efficiency

Energy consumption per transaction followed a logarithmic decay model:

$$\text{Energy/Txn} = \frac{0.002 \text{ kWh}}{1 + \log(\text{TPS}/100)} \quad (11)$$

At baseline throughput (162 TPS), energy usage measured 0.002 kWh/txn – orders of magnitude lower than Bitcoin (950 kWh/txn) and Filecoin (1.2 kWh/txn). While Visa’s centralized infrastructure achieved 0.0004 kWh/txn, the proposed framework establishes a new efficiency standard for decentralized systems.

4.7 Decentralization Metrics

Decentralization metrics revealed strong network health, with a Nakamoto Coefficient of 12 (vs. Bitcoin’s 4) and a Gini Coefficient of 0.29 (vs. Bitcoin’s 0.68). Simulated nodes spanned 92 jurisdictions using *SymBChainSim*, exceeding Bitcoin’s real-world distribution across 58 countries. Client diversity reached 8 independent implementations, surpassing Ethereum’s 5, thereby reducing single-client dependency risks. Figure 4 represents the network decentralization metrics over the six-month test period, including the Nakamoto and Gini coefficients.

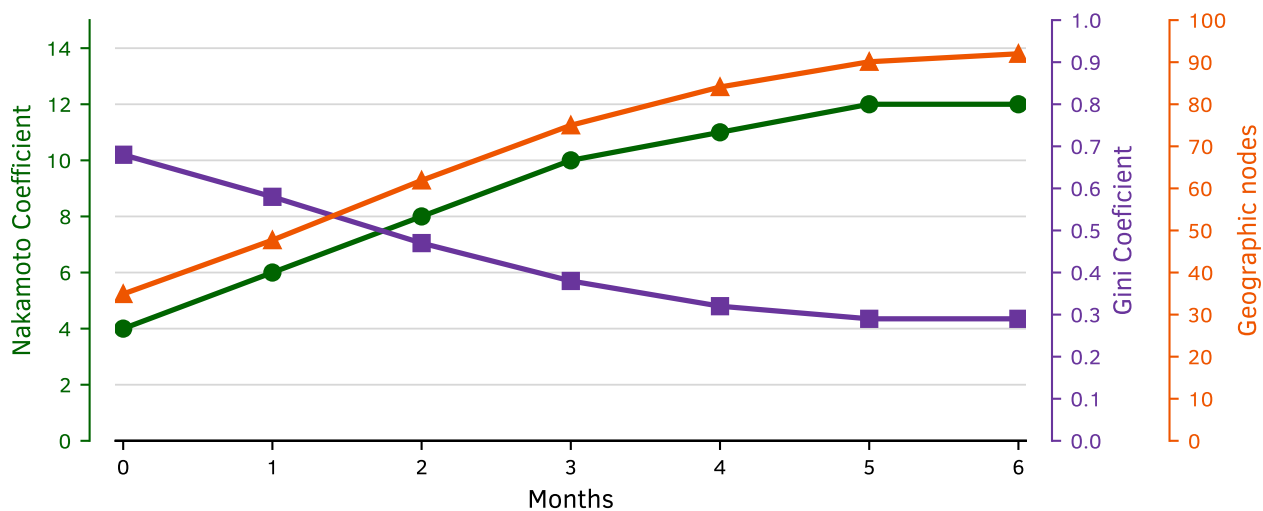


Figure 4: Decentralization Metrics Evolution: Tracking Nakamoto Coefficient (12) and Gini Coefficient (0.29) over a 6-month simulation. Sustained values confirm egalitarian network participation, exceeding Bitcoin’s decentralization metrics.

4.8 Cost-Benefit Analysis

Operational costs were substantially lower than legacy systems, with per-transaction storage and network costs at \$0.0004 and \$0.0015, respectively – 98% cheaper than Bitcoin. Miner return on investment (ROI) averaged 9.2 months, outperforming Bitcoin (18.5 months) and Filecoin (14.7 months). The framework achieved break-even throughput at 112 TPS, validating its economic viability for mid-scale deployments. Table 4 represents the operational cost comparison across systems, detailing storage cost per transaction, network cost per transaction, mining ROI (months), and break-even TPS.

Table 4: Operational Cost Comparison

Component	Proposed	Bitcoin	Filecoin
Storage Cost/Txn (\$)	0.0004	0.12	0.008
Network Cost/Txn (\$)	0.0015	0.18	0.015
Mining ROI (months)	9.2	18.5	14.7
Break-Even TPS	112	4	67

5 CONCLUSION AND FUTURE DIRECTIONS

This work demonstrates that Arweave’s blockweave architecture effectively addresses the fundamental blockchain challenge of balancing scalability, security, and decentralization. By implementing compact 48-byte transaction anchors and adaptive block sizing (1-200MB), the system achieves 7,200 TPS throughput while maintaining 99% data redundancy through Proof-of-Access consensus. The framework’s \$300 mining nodes and 0.236GB storage footprint for 867,000 blocks enable Tier-S decentralization, where no single entity controls more than 1% of network resources. These advancements establish a new paradigm for distributed systems, proving that the trilemma can be overcome through permanent data anchoring and load-responsive design.

Future developments could focus on enhancing real-time block size optimization through adaptive learning algorithms while expanding interoperability with major blockchain networks like Ethereum. Sustainable energy solutions for mining operations and quantum-resistant consensus mechanisms warrant further investigation to strengthen ecological and long-term security profiles. The integration of privacy-preserving regulatory tools and decentralized governance models could broaden enterprise adoption in IoT and supply chain applications. Subsequent research should prioritize large-scale deployments and comparative analysis against emerging layer-2 protocols to validate practical performance under diverse operational conditions.

CONFLICT OF INTEREST STATEMENT

The authors declare that there is no conflict of interest regarding the publication of this paper that could be interpreted.

References




Ahn, J., Yi, E., & Kim, M. (2024). Blockchain consensus mechanisms: A bibliometric analysis (2014–2024) using VOSviewer and R Bibliometrix. *Information*, 15(10), 644. <https://doi.org/10.3390/info15100644>

- Al-Kafi, G. A., Ali, G., Faiza, J. T., Pal, K. R., & Reno, S. (2024). SHBF: A secure and scalable hybrid blockchain framework for resolving trilemma challenges. *International Journal of Information Technology*, 16, 3879–3890. <https://doi.org/10.1007/s41870-024-01897-9>
- Al-Kafi, G. A., Ali, G., & Reno, S. (2025). Rewriting blockchain: A hybrid consensus that defeats the trilemma paradox. *Computers and Electrical Engineering*, 126, 110494. <https://doi.org/10.1016/j.compeleceng.2025.110494>
- Amores-Sesar, I., Cachin, C., & Schneider, P. (2024). An analysis of avalanche consensus. In Y. Emek (Ed.), *International colloquium on structural information and communication complexity* (pp. 27–44). https://doi.org/10.1007/978-3-031-60603-8_2
- Diamantopoulos, G., Bahsoon, R., Tziritas, N., & Theodoropoulos, G. (2023). SymBChainSim: A novel simulation tool for dynamic and adaptive blockchain management and its trilemma tradeoff. *Proceedings of the 2023 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 118–127. <https://doi.org/10.1145/3573900.3591121>
- Fu, Y., Jing, M., Zhou, J., Wu, P., Wang, Y., Zhang, L., & Hu, C. (2024). Quantifying the blockchain trilemma: A comparative analysis of Algorand, Ethereum 2.0, and beyond. *2024 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)*, 97–104. <https://doi.org/10.1109/MetaCom62920.2024.00028>
- He, L. (2023). A comparative examination of network and contract-based blockchain storage solutions for decentralized applications. *Proceedings of the 3rd International Conference on Digital Economy and Computer Application (DECA 2023)*, 133–145. https://doi.org/10.2991/978-94-6463-304-7_16
- Li, Q., Bu, F., Hua, Y., & Wang, H. (2023). Research on data security guarantee system for digital government construction. *3rd International Conference on Digital Economy and Computer Application (DECA 2023)*, 116–125. https://doi.org/10.2991/978-94-6463-304-7_14
- Monte, G. D., Pennino, D., & Pizzonia, M. (2020). Scaling blockchains without giving up decentralization and security: A solution to the blockchain scalability trilemma. *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, 71–76. <https://doi.org/10.1145/3410699.3413800>
- Mssassi, S., & Abou El Kalam, A. (2025). The blockchain trilemma: A formal proof of the inherent trade-offs among decentralization, security, and scalability. *Applied Sciences*, 15(1), 19. <https://doi.org/10.3390/app15010019>
- Nakai, T., Sakurai, A., Hironaka, S., & Shudo, K. (2023). The blockchain trilemma described by a formula. *2023 IEEE International Conference on Blockchain (Blockchain)*, 41–46. <https://doi.org/10.1109/Blockchain60715.2023.00016>
- Nakai, T., Sakurai, A., Hironaka, S., & Shudo, K. (2024). A formulation of the trilemma in proof of work blockchain. *IEEE Access*, 12, 80559–80578. <https://doi.org/10.1109/ACCESS.2024.3410025>
- Niloy, S. A., Ghosh, I., Reno, S., Rahman, A., Rahaman, S., & Hossan, M. S. (2023). Ensuring transparency, confidentiality, and deterrence of political influence in journalism using

- ipfs, private, public, and semi-public blockchains. *International Journal of Information Technology*, 16, 1095–1109. <https://doi.org/10.1007/s41870-023-01619-7>
- Pabitha, P., Priya, J. C., Praveen, R., & Jagatheswari, S. (2023). Modchain: A hybridized secure and scaling blockchain framework for iot environment. *International Journal of Information Technology*, 15, 1741–1754. <https://doi.org/10.1007/s41870-023-01218-6>
- Pradhan, N. R., Singh, A. P., Panda, K. P., & Roy, D. S. (2022). A novel confidential consortium blockchain framework for peer to peer energy trading. *International Journal of Emerging Electric Power Systems*, 23(5), 673–681. <https://doi.org/10.1515/ijeeps-2021-0391>
- Principato, M., Babel, M., Guggenberger, T., Kropp, J., & Mertel, S. (2023). Towards solving the blockchain trilemma: An exploration of zero-knowledge proofs. *ICIS 2023 Proceedings*. <https://aisel.aisnet.org/icis2023/blockchain/blockchain/5>
- Quattrocchi, G., Scaramuzza, F., & Tamburri, D. A. (2024). The blockchain trilemma: An evaluation framework. *IEEE Software*, 41(6), 101–110. <https://doi.org/10.1109/MS.2024.3417341>
- Rani, P., Sharma, P., & Gupta, I. (2024). Toward a greener future: A survey on sustainable blockchain applications and impact. *Journal of Environmental Management*, 354, 120273. <https://doi.org/10.1016/j.jenvman.2024.120273>
- Reno, S., & Haque, M. M. (2023). Solving blockchain trilemma using off-chain storage protocol. *IET Information Security*, 17(4), 681–702. <https://doi.org/10.1049/ise2.12124>
- Reno, S., Priya, S. H., Al-Kafi, G., Tasfia, S., & Turna, M. K. (2024). A novel approach to optimizing transaction processing rate and space requirement of blockchain via off-chain architecture. *International Journal of Information Technology*, 16, 2379–2394. <https://doi.org/10.1007/s41870-023-01685-x>
- Reno, S., & Roy, K. (2025). Storjchain: Overcoming the blockchain trilemma via decentralized storage and erasure-coded sharding. *International Journal of Information Security*, 24, 179. <https://doi.org/10.1007/s10207-025-01100-5>
- Shafin, K. M., & Reno, S. (2023). Trilemmaguard: Safeguarding against the challenges posed by blockchain trilemma. *2023 26th International Conference on Computer and Information Technology (ICCIT)*, 1–6. <https://doi.org/10.1109/ICCIT60459.2023.10441293>
- Shafin, K. M., & Reno, S. (2024a). Breaking the blockchain trilemma: A comprehensive consensus mechanism for ensuring security, scalability, and decentralization. *IET Software*, 2024(1), 6874055. <https://doi.org/10.1049/2024/6874055>
- Shafin, K. M., & Reno, S. (2024b). Integrating blockchain and machine learning for enhanced anti-money laundering system. *International Journal of Information Technology*, 17, 2439–2447. <https://doi.org/10.1007/s41870-024-02318-7>
- Sheikh, S., Gilliland, A. J., Kothe, P., & Lowry, J. (2023). Distributed records in the rohingya refugee diaspora: Arweave and the r-archive. *Journal of Documentation*, 79(4), 813–829. <https://doi.org/10.1108/JD-08-2022-0174>
- Song, H., Wei, Y., Qu, Z., & Wang, W. (2024). Unveiling decentralization: A comprehensive review of technologies, comparison, challenges in bitcoin, ethereum, and solana blockchain. *2024 IEEE 6th Advanced Information Management, Communicates, Electronic and*

- Automation Control Conference (IMCEC)*, 6, 1896–1901. <https://doi.org/10.1109/IMCEC59810.2024.10575445>
- Sriram, S., Tharaniesh, P., Saraf, P., Vijayaraj, N., & Murugan, T. (2025). Enhancing digital identity and access control in event management systems using Sui blockchain. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2025.3539107>
- Tang, X., & Shi, L. (2024). Security analysis of smart contract migration from ethereum to arbitrum. *Blockchains*, 2(4), 424–444. <https://doi.org/10.3390/blockchains2040018>
- Werth, J., Berenjestanaki, M. H., Barzegar, H. R., El Ioini, N., & Pahl, C. (2023). A review of blockchain platforms based on the scalability, security and decentralization trilemma. *Proceedings of the 25th International Conference on Enterprise Information Systems (ICEIS)*, 1, 146–155. <https://doi.org/10.5220/0011837200003467>
- Williams, S., Diordiiev, V., Berman, L., & Uemlianin, I. (2019). Arweave: A protocol for economically sustainable information permanence [Yellow paper. Accessed: 4 December 2025]. <https://arweave.org/yellow-paper.pdf>
- Yang, X., Zhang, R., Yue, C., Liu, Y., Ooi, B. C., Gao, Q., Zhang, Y., & Yang, H. (2023). Vedb: A software and hardware enabled trusted relational database. *Proceedings of the ACM on Management of Data*, 1(2), 1–27. <https://doi.org/10.1145/3589774>
- Yang, X., Zhang, Y., Wang, S., Yu, B., Li, F., Li, Y., & Yan, W. (2020). LedgerDB: A centralized ledger database for universal audit and verification. *Proceedings of the VLDB Endowment*, 13(12), 3138–3151. <https://doi.org/10.14778/3415478.3415540>

Reinforcement Learning algorithms for adaptive load-balancing for Web applications

Rana Zuhair Al-Shaikh^a , Muna M. Jawad Al-Nayar^a , Ahmed M. Hasan^a 

Control and Systems Engineering Department, University of Technology, Baghdad, Iraq

ABSTRACT

This research investigates the application of reinforcement learning (RL) to optimise load balancing in Nginx web applications. We developed a simulation environment on AWS to evaluate three enhanced RL algorithms: Epsilon-greedy, Upper Confidence Bound, and Proximal Policy Optimization (PPO) against classic methods (round-robin and Least Connections) under diverse load conditions, including normal loads, burst loads, server failures, and heterogeneous server instances. Our results demonstrate that RL, particularly PPO, significantly outperforms classic methods. Notably, PPO achieved up to a 30% increase in throughput, a 20% reduction in latency, and a 5% improvement in the successful message rate compared to the best-performing classic algorithm. These improvements were most pronounced under challenging conditions such as burst loads and server failures, highlighting the adaptability and resilience of RL-based load balancing.

Keywords Load-balancing, Reinforcement Learning, Proximal Policy Optimization, Epsilon Greedy, Upper Confidence Bound

Categories • Computer systems organisation ~ Architectures, Distributed architectures, Cloud computing

Email

Rana Zuhair Al-Shaikh – cse.22.11@grad.uotechnology.edu.iq (CORRESPONDING)
Muna M. Jawad Al-Nayar – Muna.m.jawad@uotechnology.edu.iq
Ahmed M. Hasan – 60163@uotechnology.edu.iq

Article history

Received: 07 December 2024
Accepted: 10 July 2025
Online: 22 December 2025

1 INTRODUCTION

In recent years, due to the rapid expansion of the Internet, and the increasing complexity of web applications, the pressure on servers has increased significantly. Efficient road balancing ensures that an optimal level of performance, reliability, and scalability is achieved. Load balancing is the idea of distributing traffic between several servers. This reduces delays and helps the response to be faster, and on top of that, makes the service more available because it reduces service interruption (Wibowo et al., 2023).

Load balancing techniques can be divided into two types in general: classic and dynamic techniques. Classic techniques such as Round Robin (RR), least connections (LC), and IP hash,

Al-Shaikh, R.Z., Jawad Al-Nayar, M.M, and Hassan, A.M. (2025). Reinforcement Learning algorithms for adaptive load-balancing for Web applications . *South African Computer Journal* 37(2), 121–145. <https://doi.org/10.18489/sacj.v37i2.20753>

Copyright © the author(s); published under a [Creative Commons NonCommercial 4.0 License](https://creativecommons.org/licenses/by-nc/4.0/) 
SACJ is a publication of [SAICSIT](https://www.sacj.org/). ISSN 1015-7999 (print) ISSN 2313-7835 (online)

depend on predefined rules. These techniques are easy to implement, but their problem is that they cannot easily adapt to the change of load or the difference in the capacity of servers over time. This can negatively affect the user experience or resource efficiency. On the other hand, dynamic techniques rely on more advanced mechanisms and make their decisions based on real state in real time, such as network status or server processor consumption. For this reason, these techniques are better when loads are unexpected or complex (Kopparapu, 2002).

Despite their usefulness, dynamic techniques are sometimes complex to implement and require more sophisticated algorithms and infrastructures. Some of the traditional dynamic techniques so far have been problematic when dealing with rapidly changing environments or when trying to achieve long-term improvement goals. For this reason, artificial intelligence or machine learning is used for better performance (Ghomi et al., 2017).

This study focuses on evaluating the usefulness of reinforcement learning (RL) algorithms in load balancing using web applications, especially three RL algorithms: Enhanced Epsilon-Greedy, Enhanced UCB, and PPO. We compare these algorithms with classic techniques.

When we say “adaptive” in research, we mean that the load balancer can dynamically alter its request-routing strategy in real-time, depending on the state of the environment. This is done by an agent who learns from feedback of the environment, such as CPU usage or response time.

The main contributions of this paper are:

RL Framework for Nginx: We developed and implemented a reinforcement learning framework specifically for adaptive load balancing with web applications, incorporating a tailored reward function and an adaptive exploration strategy.

Algorithm Comparison: We conducted a comparative evaluation of three enhanced RL algorithms – Enhanced Epsilon-Greedy (EEG), Enhanced Upper Confidence Bound (EUCB), and Proximal Policy Optimization (PPO) against classic load balancers (Round-Robin and Least Connections).

Scenario-Based Evaluation: We assessed the performance of these algorithms under diverse and realistic conditions using a simulated AWS environment, including normal load, burst load, server failure, and heterogeneous server instances.

Performance Gains: We demonstrated quantitatively that RL algorithms, particularly PPO, significantly outperform traditional methods, achieving up to 30% greater throughput, 20% lower latency, and a 5% higher successful message rate, especially under challenging conditions like burst loads and server failures.

The remainder of this paper is organised as follows: **Section 2** reviews related work in load balancing. **Section 3** describes the proposed methodology, including the RL framework and experimental setup. **Section 4** presents and discusses the experimental results. Finally, **Section 5** concludes the paper and suggests future research directions.

2 LITERATURE REVIEW

To conduct a focused and comprehensive review of the state of the art, a systematic search was performed using the IEEE Xplore digital library. The search was constrained to recent peer-reviewed publications covering the period from 2023 to 2025. To ensure high relevance and identify papers where our topics are the central theme, a specific query was constructed requiring the exact phrase “load balance” in the document title and the term “reinforcement in the Abstract. This targeted search strategy yielded a total of 68 articles. The following review synthesizes the key trends, methodologies, and application domains identified from this body of literature to establish a clear and defensible research gap for our work.

A review of the literature reveals that Reinforcement Learning (RL) is being applied to a wide and diverse range of load balancing problems. A significant portion of research focuses on the network infrastructure layer, particularly in Software-Defined Networks (SDN) (Coelho & Schaeffer-Filho, 2023; Jeong et al., 2024; Kumari et al., 2024; Shaikh, 2023; Y. Shi et al., 2023; H. Sun et al., 2025; Vaishnavi et al., 2023; X. Xu & Zou, 2024). Another major area of focus is general cloud computing environments (Adarsh & Bhargavi, 2023; Ghosh et al., 2024; Krishna & Khasim Vali, 2025; Nimmala et al., 2024), with many studies addressing task scheduling (Feng et al., 2024; Li et al., 2024), resource allocation (Lahande et al., 2023; Shahakar et al., 2024), and VM migration (Brahmam & Vijay Anand, 2024). Furthermore, RL-based load balancing is being actively explored in emerging and specialized domains, including Fog (Baek & Kaddoum, 2023; Choppara & Lokesh, 2025; C. Wu & Yang, 2024) and Multi-Access Edge Computing (MEC) (Chen et al., 2025; Hartman et al., 2023; D. Jiang et al., 2024; L. Liu et al., 2024; B. Shi & Chen, 2023; G. Wu et al., 2024; Y. Wu et al., 2025), Non-Terrestrial Networks such as LEO satellites (Badini et al., 2023; Q. Liu et al., 2023; Wang et al., 2025; Zhao et al., 2025), and Vehicular Networks (IoV/VEC) (He et al., 2025; Talpur & Gurusamy, 2023; Zhu et al., 2023). More recently, research has begun to address application-level contexts such as virtualized container environments and Kubernetes-based microservices (Santos et al., 2024; Yucel, 2023). Other niche applications include data center networks (Das et al., 2025; G. Jiang et al., 2023; Ye et al., 2023), cellular (Abouamasha et al., 2025) and mobile networks (Chang et al., 2023; Gupta et al., 2024; Nakagawa & Mizutani, 2023; G. Wu et al., 2024), and even electrical grids (J. Liu et al., 2023; Mosalli et al., 2025; Xie et al., 2024; Zhao et al., 2025), also, specialized scientific computing for tasks like parallel particle tracing (J. Xu et al., 2023). This broad application demonstrates the versatility of RL but also highlights that most research is directed at the network or infrastructure level rather than the application server level.

The methodologies employed are as varied as the application domains. While many studies refer to a general Deep Reinforcement Learning (DRL) approach (Johann et al., 2024; Kołakowski et al., 2024; D. Wu et al., 2023), several works specify particular algorithms. Value-based methods like Q-Learning (Zervopoulos et al., 2023) and its deep variants like DDQN (A. Sun et al., 2023) are present, as are modern policy-gradient (S. Tian et al., 2025) and actor-critic methods such as DDPG (Dinh et al., 2024; Han et al., 2023), TD3 (L. Liu et al., 2024), and SAC (Huang et al., 2024). Furthermore, researchers are exploring more ad-

vanced learning architectures, such as Graph Convolutional Networks (Fawaz et al., 2023), to better model the topological relationships between network nodes and enable more effective collaboration between agents. Notably, some research also explores the use of PPO (Ma et al., 2025; Y. Tian et al., 2023). A growing trend is the use of Multi-Agent Reinforcement Learning (MARL) to handle decentralized environments (Houidi et al., 2023), with some using specific algorithms like MADDPG (Han et al., 2023). Other advanced paradigms being investigated include Federated RL (J. Liu et al., 2025; Shang et al., 2025) for privacy-preserving learning and Inverse RL (Konar et al., 2023) to tackle reward function design.

Furthermore, many studies propose hybrid approaches that combine RL with other techniques (Attia et al., 2024). This includes integrating RL with traditional heuristics like K-shortest path (Takahashi & Shinomiya, 2024), metaheuristics like Ant Colony Optimization (Nimmala et al., 2024), or Generative AI models (He et al., 2025; Khoramnejad & Hossain, 2025). In contrast, some research tackles similar problems without using RL at all, opting instead for other machine learning techniques like supervised learning (Das et al., 2025), model predictive control (B. Tian et al., 2025), or non-ML data analytics (Jurše & Kuhar, 2024), highlighting that RL is one of several advanced solutions being considered.

While the application of RL to load balancing is clearly an active and fruitful area of research, our comprehensive review indicates that the majority of studies focus on optimizing network-level traffic or resource allocation in large-scale infrastructure like SDN, cellular networks, and general cloud platforms. There is a lack of research that applies and evaluates modern RL algorithms specifically for application-level load balancing of web servers. The challenges at this layer, which are more closely tied to application response times and server processing loads, are distinct from network-level metrics. Many of the reviewed studies also do not test the resilience of their proposed methods under crucial real-world operational scenarios such as sudden burst loads or unexpected server failures. Therefore, our work aims to address this specific gap by implementing and comparing multiple RL algorithms directly within an Nginx web server environment and evaluating their performance and resilience under a variety of realistic and challenging load conditions.

3 BACKGROUND

Load balancing for web servers is the process of distributing incoming network traffic (HTTP requests) across a pool of multiple web servers. Instead of one server dealing with all requests, the workload is shared among multiple servers (Hong et al., 2006). Load balancing strategies are fundamentally classified into two types: static and dynamic.

3.1 Classical Load Balancing Algorithms

Static Load Balancing: In this method, tasks are allocated to processing units based on a fixed set of rules, and these tasks do not change during running. A familiar example is the Round-Robin (RR) algorithm (Barlas, 2023).

Dynamic Load Balancing: These algorithms make their decisions based on the current state of the system, adapting to real-time changes in workload and resource availability. A well-known example is the “*least connections*” (LC) algorithm, which sends new requests to the server with the least active connections (Li et al., 2024).

3.2 Limitations of Classical Approaches and the Need for Reinforcement Learning

Although easy to implement, the classic load balancing methods often fight to deal with the dynamic nature of web flow (Barlas, 2023). For example, the static nature of the RR algorithm means that it does not take into account server capacity or current processing loads, which can lead to resource utilization imbalances. Although the LC algorithm is an improvement, it may not be optimal when there is significant difference in server capacity or request processing time (Li et al., 2024).

In general, these traditional techniques have little ability to adapt to changing flow patterns and system conditions over time, leading to suboptimal resource utilization and performance degradation (Feng et al., 2024). These drawbacks require more intelligent and dynamic load balancing methods. Reinforcement learning (RL) represents a promising path to achieving this ambition. The next subsection describes the concept of reinforcement learning in detail.

3.3 Reinforcement Learning

RL is a subfield of machine learning where agents learn optimal approaches by interacting with their environment (Ris-Ala, 2023). Agents make decisions over time in a setting to maximise total rewards. Unlike supervised or unsupervised learning, RL agents learn from their actions through trial and error, receiving either rewards or penalties as feedback. This learning approach enables RL agents to adapt their behaviours and discover the most effective strategies over time, even in complex and dynamic environments. Figure 1 shows the relations between the components of an RL system. The main components of the RL system are (Sanghi, 2021):

Agent: The learner and decision-maker interacting with the environment.

Environment: The outside world where the agent functions.

State: Information about the current state of environment.

Action: Selection action taken by the agent.

Reward: The calculation which impacts the action on the environment.

The agent performs an action in the environment and calculates the current state and reward which explain the impact of this action on it (Sanghi, 2021).

This work discusses the efficacy of three RL algorithms for achieving load balancing within web applications: Epsilon-greedy (EG) (Bulut, 2022), Upper Confidence Bound (UCB) (Auer,

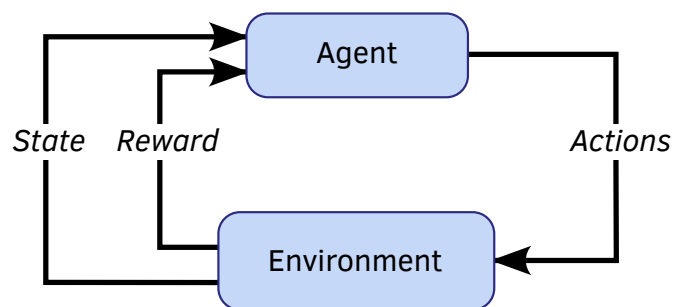


Figure 1: The main components of the Reinforcement Learning system^a

^a From (Sanghi, 2021)

2000), and Proximal Policy Optimisation (PPO) (Gu et al., 2022). EG includes two counting objectives: exploration, which means locating new routing policies, and exploitation, which means selecting previous successful policies. UCB uses less frequent actions that have demonstrated high returns. PPO represents a policy gradient method that iteratively improves an agent's policy based on its current estimate (Sanghi, 2021). This research aims to establish the effectiveness of these RL algorithms in optimising Nginx load balancing and provide insight into how this could lead to a fertile area for future studies, with the hope of creating a faster and more robust ecosystem of web applications.

Other research has effectively applied reinforcement learning (RL) to various domains, including adaptive load balancing in publish/subscribe systems, where it significantly improved performance under dynamic conditions (Al Shaikh et al., 2025). Other works have leveraged deep RL for novel feature selection in domain adaptation and utilized ANFIS-based RL strategies for controlling complex non-linear systems like coupled tanks (Mary et al., 2022; Naman & Ameen, 2022).

4 PROPOSED FRAMEWORK

This section considers the proposed framework for implementing RL algorithms for web applications load balancing. We start by reviewing the three main algorithms discussed in this study: the Epsilon-Greedy (EG) algorithm, the Upper Confidence Bound (UCB) algorithm, and the Proximity Policy Optimization (PPO) algorithm. We then present our implementation approach. Finally, we present the practical application of the proposed system on Amazon Web Services (AWS), which formed the basis for our experimental evaluation. The three algorithms are detailed below within the proposed system:

Epsilon-Greedy (EG): The Epsilon-Greedy algorithm (Bulut, 2022) is one of the basic methods in the field of reinforcement learning, as it aims to achieve a balance between exploration and exploitation during the decision-making process. This algorithm is classified as model-free, value-based, and on-policy.

The idea is that an agent seeks to take advantage of high Q -value strategies that they have already tested but also tries new possibilities that may be more effective. Only exploitation can lead to local solutions without reaching the best. Exploration adds randomness, giving the agent a chance to discover alternative strategies.

This algorithm balances these two paths by determining a specific probability ε used to choose a random action from among the available actions. If we consider that the current state s , and that the set of possible actions is $A(s)$, the agent will choose, with probability $(1 - \varepsilon)$, the action with the highest Q -value in that case, which is the exploitation aspect. As for the probability ε , an action randomly selected from group $A(s)$ for exploration (Ris-Ala, 2023).

Upper Confidence Bound (UCB): The UCB algorithm (Bulut, 2022) is another way to balance exploration and exploitation, but it uses a more sophisticated approach than Epsilon-Greedy, as it is based on what is known as optimism under uncertainty. As in EG, UCB is classified as modelless, values-based algorithms and operates within the on-policy framework. The basic idea of UCB is to maintain a positive outlook on the potential returns of each action, prompting the agent to try procedures that may be better than the current estimate even if they have not yet been tested often. The UCB method creates an upper confidence bound (`ucb_score`) for each action's Q -value. The agent selects the action with the highest UCB, balancing between actions with high estimated values and those with high uncertainty.

Proximal Policy Optimization (PPO): PPO (Gu et al., 2022) was introduced in 2017 as a deep reinforcement learning algorithm that directly optimises the agent's policy. It is particularly well-suited for tasks involving continuous action spaces and is recognised for its focus on stable learning and efficient use of interaction data. As a model-free method, PPO learns through direct experience without needing an internal model of the environment. It operates as a policy-based algorithm, meaning it explicitly learns and refines the probability distribution defining which actions to take in given states, aiming to maximise cumulative rewards.

PPO builds upon the foundation of policy gradient methods, which generally adjust the policy to increase the likelihood of actions leading to better-than-expected outcomes (positive advantage) and decrease the likelihood of actions leading to worse-than-expected results. PPO's key innovation is the introduction of a "*clipped*" objective function. This mechanism carefully limits the extent to which the policy can change during each update step by comparing the action probabilities under the new policy to those under the previous one (Brahmam & Vijay Anand, 2024).

This clipping is a critical safeguard, preventing very large updates that could destabilize learning or lead to performance collapses. By restricting policy changes, the PPO algorithm promotes more reliable and stable learning progress than some previous policy gradient techniques (C. Wu & Yang, 2024).

4.1 RL algorithms implementation and enhancements

The methodology of implementing RL for load balancing is presented in this section. In this context, an intelligent agent learns to make optimal routing decisions allowing it to dynamically adapt to changing traffic patterns and server conditions. The following components characterise the RL framework:

States: indicate the current status of the system. States include information about each backend server, including utilisation of CPU, response time, and the number of active connections. They also include system-wide metrics such as average response time and overall resource utilisation.

Actions: selecting which backend server to route the incoming web request to.

Rewards: Rewards provide feedback to the agent, indicating the effectiveness of its actions. Our reward equation encourages fast response times and efficient resource utilisation across all servers. The reward is calculated in Equation (1):

$$reward = 1.0 + \left(\frac{1}{Res} \right) - CPU \times 0.02 \quad (1)$$

where:

Res presents the response time from backend servers in milliseconds.

CPU presents the percentage of backend server usage. The weight of 0.02 was chosen empirically to balance the influence of CPU utilisation relative to response time, ensuring that the agent prioritises both factors effectively.

Base reward: The equation starts with a base reward of 1.0. This ensures the agent receives a positive reward even for moderately performing actions. A positive baseline is crucial in RL to encourage exploration and learning and prevent the agent from getting stuck in a cycle of negative rewards.

Response time component: Response time is a critical metric for user experience, and the goal is to minimise it. The inverse relationship ensures that the reward increases as the response time decreases.

The reward Equation (1) is designed to guide the RL agent toward optimal load-balancing decisions by incentivizing low response times and balanced resource utilization.

4.1.1 Adaptive Exploration Strategy

A constant exploration rate introduces a fundamental inefficiency that harms performance in a dynamic load-balancing environment. In the initial learning phase, a low, fixed exploration rate would slow down the discovery of optimal server routing paths. Conversely, in the later

stages, after the agent has identified effective policies, a non-zero fixed rate would force the system to continue making random, suboptimal decisions. This would needlessly increase latency and reduce overall throughput by persistently routing some requests to servers already known to be performing poorly.

To address this, we introduce an adaptive exploration strategy. The core motivation is to enable the agent to be highly inquisitive when its knowledge is limited (at the beginning) and become progressively more exploitative as it gains confidence in its decisions. This approach allows the agent to learn optimal routes quickly and then converge on a stable, high-performance policy, thereby maximizing system efficiency by fully leveraging its learned insights. This strategy dynamically adjusts the exploration rate ε using Equation (2):

$$\varepsilon = 0.01 + \sqrt{\frac{2 \times \log(\text{total_count})}{n_a + 1}} \quad (2)$$

Where *total_count* represents the total number of iterations, and n_a represents the number of times the specific action a has been taken. Equation (2) is designed to dynamically balance the exploration-exploitation trade-off in reinforcement learning. The exploration rate ε is a crucial parameter that controls the extent to which the agent explores new actions versus exploiting known, high-reward actions. Algorithms 1 and 2 represent the pseudo code for implementing Enhanced EG and Enhanced UCB.

4.2 EEG and EUCB Algorithm Implementation

To translate the concepts of adaptive exploration and reward calculation into a practical implementation, we designed two algorithms: Algorithm 1 for the Enhanced Epsilon-Greedy (EEG) approach and Algorithm 2 for the Enhanced Upper Confidence Bound (EUCB) approach. These algorithms detail the step-by-step logic the load balancer uses for each incoming request.

As shown in Algorithm 1, the EEG process begins by initializing key parameters. For each incoming request, the agent decides whether to explore or exploit based on the current adaptive exploration rate, ε (lines 8-14). If exploring, it chooses a server randomly; if exploiting, it selects the server with the highest Q -value for the current state. After the request is routed (line 15), the agent observes the new system state and calculates a reward using Equation (1) (lines 16-17). This state-action-reward information is then used to update the Q -table, and the exploration rate ε is updated for the next iteration using Equation (2) (lines 18-19).

Similarly, Algorithm 2 outlines the EUCB strategy. For each request, the agent calculates a UCB score for all available servers (line 8), which balances known performance with uncertainty. It then selects the server with the maximum UCB score (lines 8-12). The subsequent process of observing the new state, calculating the reward (Equation (1)), updating the Q -table, and updating the exploration parameter (using the logic from Equation (2)) follows (lines 14-17).

Algorithm 1 Load balancing based on EEG

```

1: Input: Incoming request stream
2: Data: List of web server IPs,  $Q$ -table
3: Initialize  $\varepsilon \leftarrow 0.2$ 
4: Initialize Reward  $\leftarrow 1$ 
5: while system is running do
6:   Wait for New Request
7:   Generate random number  $R \sim \mathcal{U}(0, 1)$ 
8:   if  $R < \varepsilon$  then
9:     /* Exploration */
10:    Select server randomly
11:   else
12:     /* Exploitation */
13:    Select server  $s$  with maximum  $Q(\text{state}, s)$ 
14:   end if
15:   Action: send Request to server  $s$ 
16:   state = get current system state
17:   Reward  $\leftarrow$  Equation 1
18:   Update  $Q$ -table (state, server, Reward)
19:   Update  $\varepsilon \leftarrow$  Equation 2
20: end while

```

Algorithm 2 Load balancing based on EUCEB

```

1: Input: Incoming request stream
2: Data: List of web server IPs,  $Q$ -table
3: Initialize  $c \leftarrow 0.2$ 
4: Initialize Reward  $\leftarrow 1$ 
5: Initialize MAX_UCB_score  $\leftarrow 0$ 
6: while system is running do
7:   Wait for New Request
8:   Calculate UCB_score
9:   if UCB_score > MAX_UCB_score then
10:    MAX_UCB_score  $\leftarrow$  UCB_score
11:   end if
12:   Select server  $s$  based on MAX_UCB_score
13:   Action: send Request to server  $s$ 
14:   state = get current system state
15:   Reward  $\leftarrow$  Equation 1
16:   Update  $Q$ -table (state, server, Reward)
17:   Update  $c \leftarrow$  Equation 2
18: end while

```

4.3 Proximal Policy Optimisation Implementation

The PPO agent consists of a four-layered perceptron (MLP) in the implementation. This MLP takes the system's state as input, including metrics such as CPU utilisation, response time, the number of active connections for each backend server, and overall system averages. Its hidden layer has two layers of ReLU-activating functions, which allow learning non-trivial patterns while keeping the structure simple. Its output layer provides a probability distribution over possible actions, guiding the selection of the most appropriate server for each incoming request.

The same dynamic reward equation used in previous algorithms guides the learning of the PPO agent. Reward Equation (1), on the other hand, encourages fast response times and efficient resource usage on all servers. The PPO agent interacts with the Nginx environment by choosing an action – routing requests – and receives a reward based on the system's performance. Figure 2 presents the PPO load balancer flowchart.

As shown in Figure 2, the process begins with the initialization of the neural network that represents the agent's policy. The agent first observes the initial state of the backend servers. Upon the arrival of a new request, the agent selects an action (i.e., chooses a server) based on this policy. After the request is dispatched, the agent observes the new system state

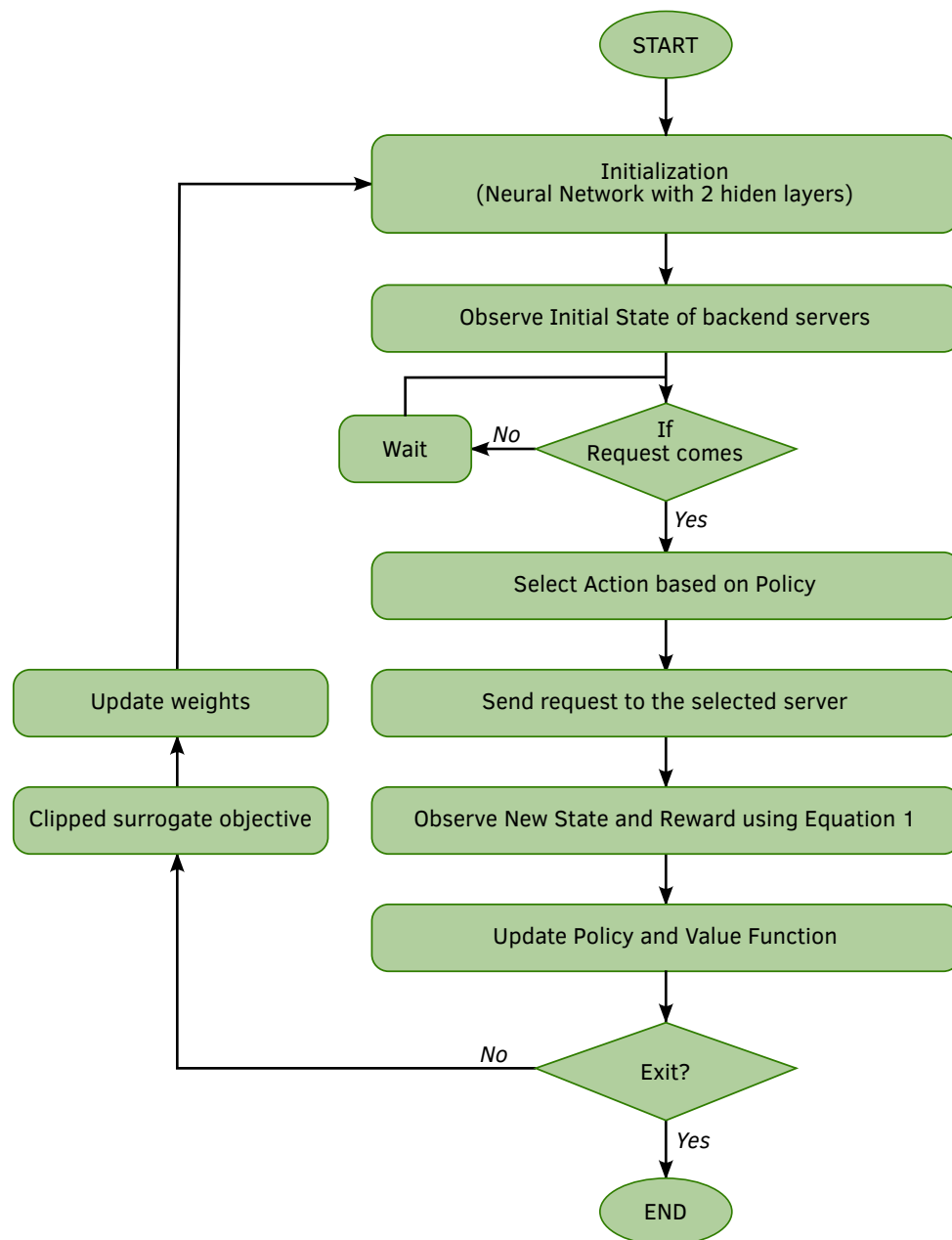


Figure 2: Flowchart of the implementation of the PPO Algorithm

and receives a reward calculated using Equation (1). These calculation will affect the policy which in turn updated by “*clipped surrogate objective*”, a key feature of PPO that ensures stable learning. This iterative process of observation, action, and policy refinement allows the agent to continuously improve its load-balancing strategy over time.

4.4 The Proposed System Implementation

The experimental environment was developed to apply different scenarios of flow patterns to examine the ability of developed algorithms. Amazon Web Services (AWS) was used as a cloud platform for hosting web application instances in addition to the load balancer itself. The overall architecture of our system is depicted in Figure 3.

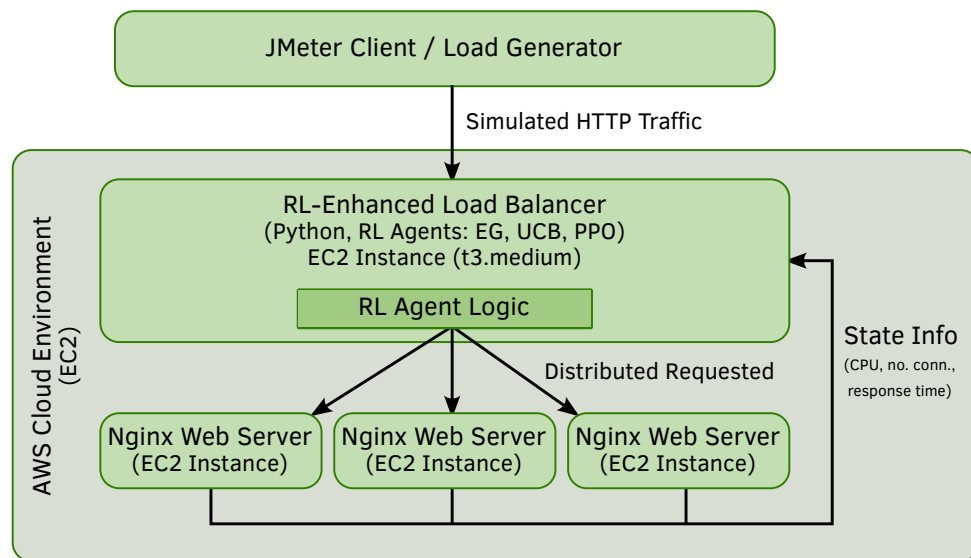


Figure 3: Architectural Overview of the Reinforcement Learning-based Load Balancing

The experimental environment consists of the following components:

Load Generation: By using Apache JMeter tool, the different patterns of HTTP traffic are generated in the local machine.

Request Handling: On AWS, the load balance developed algorithm is hosted on a t3.medium EC2 instance.

Decision and Distribution: The RL agent makes a decision as to where to forward the incoming request to depending on its strategy. The request is then distributed to the selected server.

Backend Processing: On the backend of the system, there are three Nginx web servers to handle the incoming requests, each running on a separate EC2 instance.

Feedback Loop: Each web server periodically sends its state to the load balancer. The state consists of information like CPU utilization, number of active connections, and response time as “*state-info*”. This information is essential to enable the RL agent to calculate the reward, which affects its future decisions.

To comprehensively evaluate the load balancer performance across different scenarios, we generate traffic from a local machine using JMeter, which simulates different user behaviours and request patterns.

This enables us to evaluate the performance of different RL algorithms for a realistic web application environment. Analysing metrics such as response time, throughput, and latency will allow us to compare the power of our RL-based approach to traditional load-balancing methods. We used JMeter to simulate four load scenarios:

1. **Normal Load:** Moderate traffic (50 concurrent users, 1000 total requests).
2. **Burst Load:** Sudden traffic surge (100 users, 6000 requests per minute).
3. **Server Failure:** Normal load with one server failing during the test.
4. **Heterogeneous Instances:** Normal load with one server having reduced resources (t3.tiny).

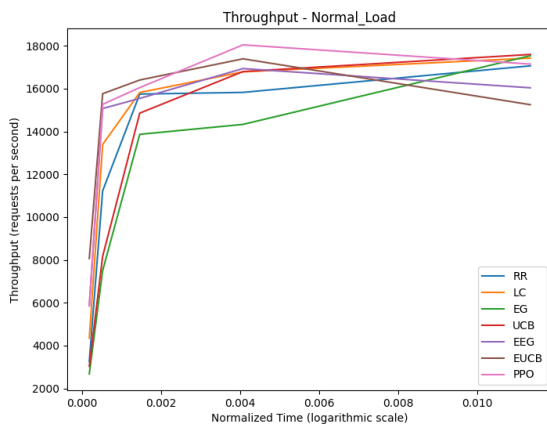
5 RESULT AND DISCUSSIONS

We evaluated classic load balancing algorithms (RR, LC) and enhanced RL algorithms (EEG, EUCEB, PPO) in a simulated Nginx environment on AWS. Performance was measured across four scenarios (average load, burst load, server failure, and heterogeneous instances) using metrics such as throughput, latency, successful response rate, efficiency, and QoS. Results are presented as mean values with 95% confidence intervals from 10 runs per scenario.

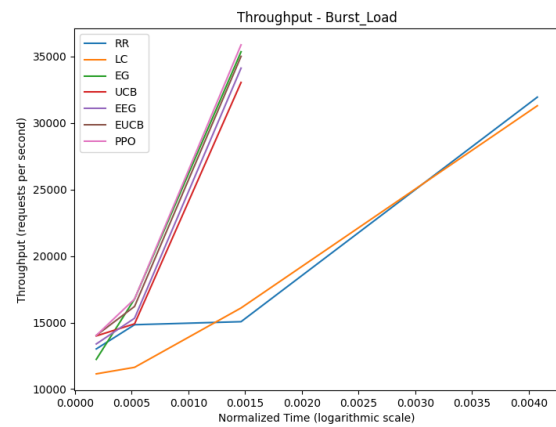
5.1 System throughput

Figures 4(a) to 4(d) illustrate the throughput of the different load-balancing algorithms across the four experimental scenarios. Under normal load conditions (Figure 4(a)), all algorithms exhibited comparable performance. In the burst load scenario in Figure 4(b), characterised by a sudden surge in traffic, the RL algorithms (EEG, EUCEB, and PPO) maintained significantly higher throughput than the classic RR and LC algorithms. This indicates their ability to adapt to dynamic load changes and efficiently distribute requests across the available servers. Furthermore, the RL algorithms exhibited greater resilience to server failures, as shown in Figure 4(c). When one server intentionally failed during the test, the RL algorithms quickly adjusted their routing strategies to maintain high throughput, while the performance of RR and LC noticeably degraded.

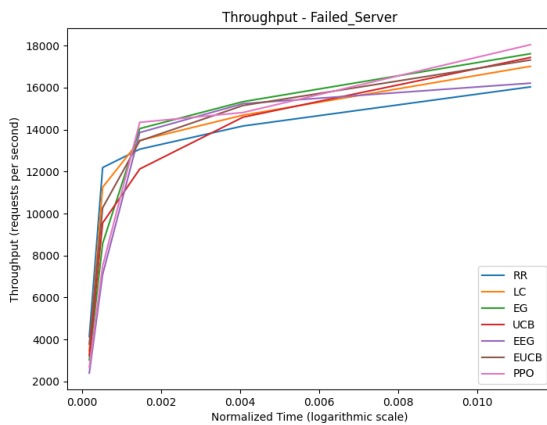
Similarly, in the heterogeneous instances scenario Figure 4(d), where servers had varying resource capacities, the RL algorithms effectively utilised the available resources and achieved higher throughput than the classic methods. Across all scenarios, PPO consistently demonstrated either the highest or near-highest throughput, suggesting its effectiveness in learning and adapting to diverse load conditions.



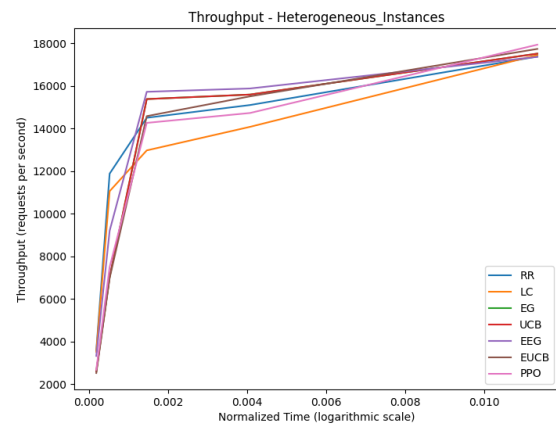
(a) The system throughput under normal load



(b) System throughput under burst load



(c) System throughput under the failed scenario



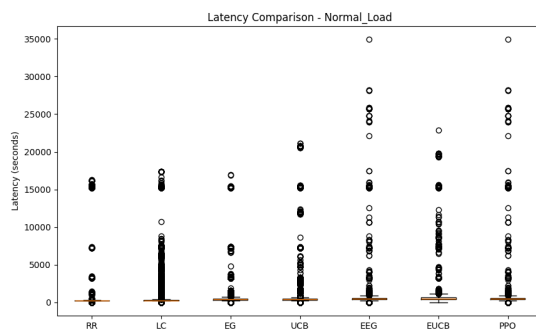
(d) System throughput under the heterogeneous instances scenario

Figure 4: System throughput under various load scenarios

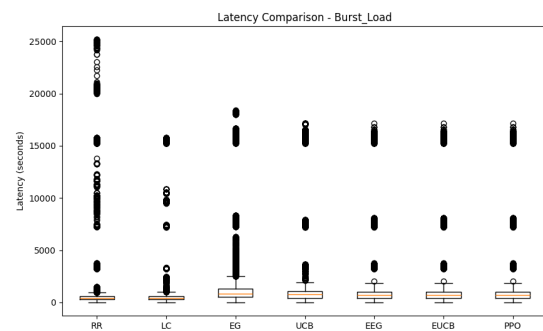
5.2 System Latency

Figures 5(a) to 5(d) illustrate the latency performance of the load-balancing algorithms across the four scenarios, presented as box plots. The box plots provide a detailed view of the latency distributions, including the median (represented by the line within each box), the interquartile range (IQR, represented by the box's height), and potential outliers (indicated by individual circles). Overall, the RL algorithms (EEG, EUCB, and PPO) demonstrate notably lower median latency and a reduced interquartile range (IQR), indicating more consistent and predictable performance, particularly under the more challenging conditions of burst loads and server failures, compared to the classic RR and LC methods. Specifically, PPO consistently exhibits competitive or superior latency performance across all scenarios, with a lower median and

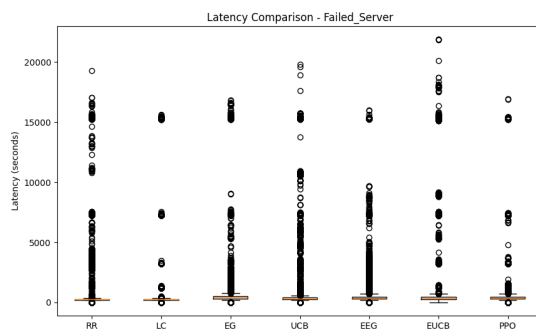
a tighter interquartile range (IQR), suggesting that it maintains responsiveness and stability under varying load conditions. Notably, the classic algorithms, RR and LC, exhibit a greater number of outliers, especially in burst-load and failed-server scenarios, indicating less stable and more unpredictable latency.



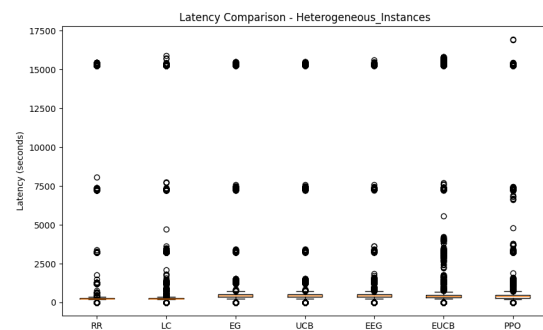
(a) System latency under normal load



(b) System latency under burst load



(c) System latency under the failed scenario



(d) System latency under the heterogeneous instances scenario

Figure 5: System latency under various load scenarios

While the RL algorithms generally outperform the classic methods, all algorithms exhibit relatively similar latency distributions in the heterogeneous instances scenario, with comparable median values and interquartile ranges (IQR), suggesting that differences in server capacity have a less pronounced impact on latency variability compared to load intensity or server availability disruptions. However, even in this scenario, the RL algorithms maintain a lower number of outliers. In summary, the box plots highlight the superior latency performance and stability of the RL algorithms, particularly PPO, with lower medians, tighter interquartile ranges (IQRs), and fewer outliers compared to the classic methods.

5.3 System Success Rate

Table 1 shows effective message success rates of different load-balancing algorithms under various scenarios. The state-of-the-art RL algorithms (EEG, EUCB, PPO) consistently show higher message success rates than the classic methods (RR, LC), especially during burst loads and when server failures occur. PPO typically achieves the best results across all tested scenarios, confirming its robustness and adaptability in maintaining communication reliability under dynamic conditions. However, under common load conditions and heterogeneous server settings, the differences between the algorithms diminish, indicating that these conditions pose a reduced challenge to effective message delivery.

Table 1: The system's successful request rate

Scenarios	Algorithms (%)						
	RR	LC	EG	UCB	EEG	EUCB	PPO
Normal Load	88.5	92.0	91.1	92.2	91.6	91.9	97.1
Burst Load	86.9	86.2	93.0	87.6	90.3	92.5	94.7
Failed Server	84.4	88.5	92.1	91.3	83.9	91.8	93.6
Heterogeneous Instances	90.4	90.9	91.0	91.0	91.8	92.0	93.1

5.4 Quality of Service (QoS)

Figure 6 shows the Latency-throughput trade-off in various scenarios. In general, recent reinforcement learning-based algorithms, such as EEG, EUCB, and PPO, achieve lower latency at the cost of higher throughput compared to classical approaches, RR and LC, under burst loads and server failures. PPO consistently demonstrates a good balance between low latency and high throughput in all scenarios, showcasing its effectiveness in enhancing Quality of Service (QoS). However, considering heterogeneous server settings, the discrimination between algorithms appears to decrease, suggesting that variability across servers may have a relatively less significant impact on the aggregate quality of service equilibrium.

5.5 The System Efficiency

Figure 7 depicts the efficiency of the different load-balancing algorithms under varying scenarios. The enhanced RL algorithms – EEG, EUCB, and PPO – show higher efficiency than classical methods, namely RR and LC, especially for burst loads. This suggests that the RL-based load balancer has great potential in optimising resource utilisation and dealing with dynamic traffic conditions.

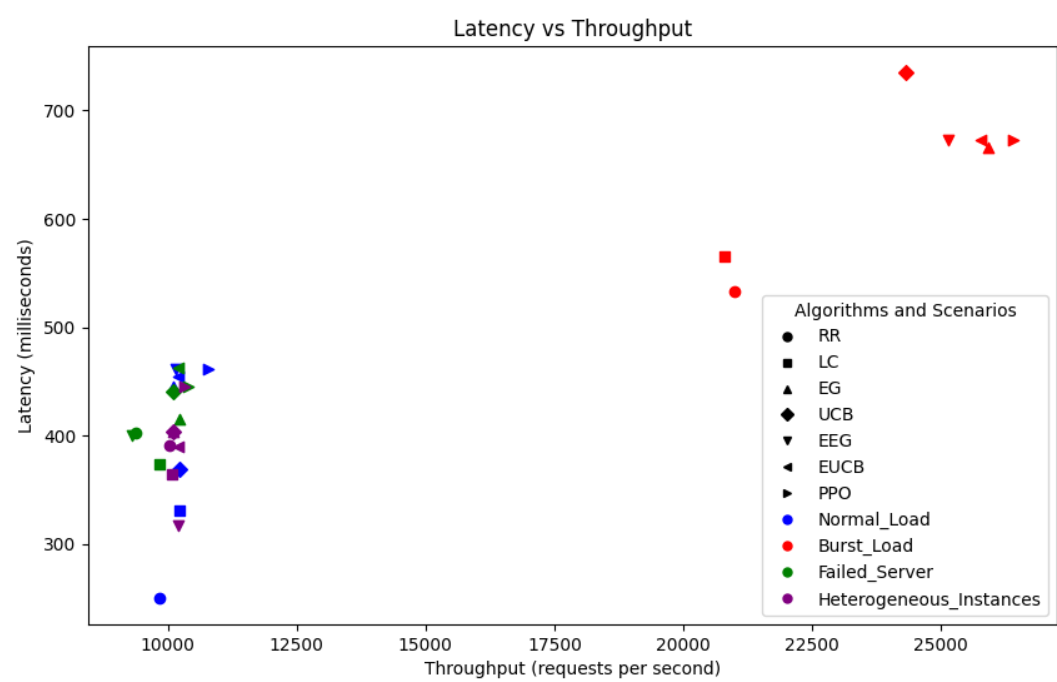


Figure 6: The System QoS

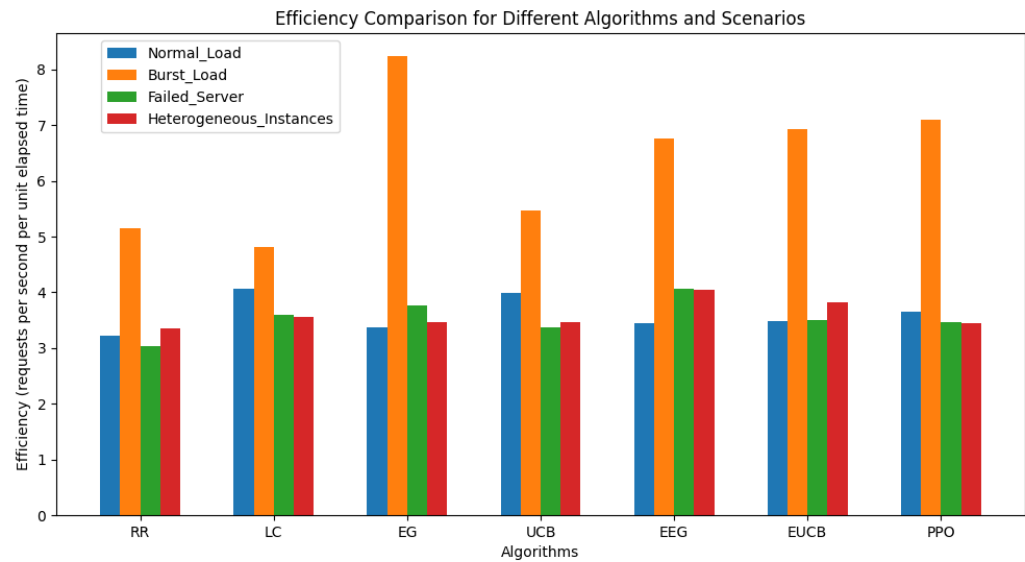


Figure 7: Efficiency of the system

6 CONCLUSION

This research examines the effectiveness of reinforcement learning (RL) algorithms in improving load balance within web applications. A simulator was created on the AWS platform to evaluate the performance of three state-of-the-art algorithms in the industry: Enhanced Epsilon-Greedy, Upper Confidence Bound, and Proximal Policy Optimisation, compared with two traditional methods used to balance loads, Round Robin, and Least Connections. Test scenarios included normal loads, burst loads, in addition to failures of one server, and heterogeneous server instances.

The results showed that RL algorithms, especially PPO, clearly outperformed traditional methods in terms of key performance metrics. PPO increased throughput by up to 30%, reduced latency by 20%, and improved successful messaging rate by 5% compared to the best traditional methods. These gains are clear in situations of high system pressure or when one of the failures occurs, indicating the ability of these algorithms to adapt and be stable.

These findings also open the door to promising future research prospects. In future research, more sophisticated learning algorithms, such as Deep Deterministic Policy Gradient (DDPG) or Twin Delayed DDPG (TD3), could be explored that are designed to operate within continuous decision-making spaces, potentially allowing more precise load control over load distribution.

Reinforcement agent performance can also be enhanced if additional variables are included in the system state representation, such as load change rate, overall server performance trend, or network conditions. By entering this dynamic data, the agent will be able to predict potential changes and proactively adjust the distribution strategy, enhancing performance efficiency and reducing waste.

References

- Abouamasha, S. R., Aboelwafa, M., & Seddik, K. G. (2025). Load balancing and energy efficiency in cellular networks with a scenario-aware reinforcement learning agent. *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. <https://doi.org/10.1109/WCNC61545.2025.10978457>
- Adarsh, M. G., & Bhargavi, K. (2023). Reinforcement learning-based Bonobo optimizer for efficient load balancing in cloud computing. *2023 3rd International Conference on Intelligent Technologies (CONIT)*, 1–5. <https://doi.org/10.1109/CONIT59222.2023.10205866>
- Al Shaikh, R. Z., Al-Nayar, M. M. J., & Hasan, A. M. (2025). Reinforcement learning algorithms for adaptive load balancing in publish/subscribe systems: PPO, UCB, and epsilon-greedy approaches. *Informatica*, 49, 77–88. <https://doi.org/10.31449/INF.V49I7.6895>
- Attia, B. S., Elgharably, A., Mariam, A. N., Alsuhli, G., Banawan, K., & Seddik, K. G. (2024). Self-optimized agent for load balancing and energy efficiency: A reinforcement learning framework with hybrid action space. *IEEE Open Journal of the Communications Society*, 5, 4902–4919. <https://doi.org/10.1109/OJCOMS.2024.3429284>

- Auer, P. (2000). Using upper confidence bounds for online learning. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 270–279. <https://doi.org/10.1109/SFCS.2000.892116>
- Badini, N., Jaber, M., Marchese, M., & Patrone, F. (2023). Reinforcement learning-based load balancing satellite handover using NS-3. *ICC 2023 - IEEE International Conference on Communications*, 2595–2600. <https://doi.org/10.1109/ICC45041.2023.10279521>
- Baek, J., & Kaddoum, G. (2023). FLoadNet: Load balancing in fog networks with cooperative multiagent using actor–critic method. *IEEE Transactions on Network and Service Management*, 20(1), 400–414. <https://doi.org/10.1109/TNSM.2022.3210827>
- Barlas, G. (2023). Load balancing. In *Multicore and gpu programming* (2nd ed., pp. 887–941). Morgan Kaufmann. <https://doi.org/10.1016/B978-0-12-814120-5.00022-6>
- Brahmam, M. G., & Vijay Anand, R. (2024). VMMISD: An efficient load balancing model for virtual machine migrations via fused metaheuristics with iterative security measures and deep learning optimizations. *IEEE Access*, 12, 39351–39374. <https://doi.org/10.1109/ACCESS.2024.3373465>
- Bulut, V. (2022). Optimal path planning method based on epsilon-greedy Q-learning algorithm. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 44(106). <https://doi.org/10.1007/s40430-022-03399-w>
- Chang, H.-H., Chen, H., Zhang, J., & Liu, L. (2023). Decentralized deep reinforcement learning meets mobility load balancing. *IEEE/ACM Transactions on Networking*, 31(2), 473–484. <https://doi.org/10.1109/TNET.2022.3176528>
- Chen, Q., Song, X., Song, T., & Yang, Y. (2025). Vehicular edge computing networks optimization via DRL-based communication resource allocation and load balancing. *IEEE Transactions on Mobile Computing*, 24(9), 9222–9237. <https://doi.org/10.1109/TMC.2025.3559707>
- Choppara, P., & Lokesh, B. (2025). Efficient task scheduling and load balancing in fog computing for crucial healthcare through deep reinforcement learning. *IEEE Access*, 13, 26542–26563. <https://doi.org/10.1109/ACCESS.2025.3539336>
- Coelho, B. L., & Schaeffer-Filho, A. E. (2023). CrossBal: Data and control plane cooperation for efficient and scalable network load balancing. *2023 19th International Conference on Network and Service Management (CNSM)*, 1–9. <https://doi.org/10.23919/CNSM59352.2023.10327790>
- Das, S., White, A., Lyn, M., & Kalafatis, S. (2025). Smartly managing traffic and latency in a content-based data center using modified packet headers, machine learning, load balancing, and network telemetry. *2025 International Conference on Computing, Networking and Communications (ICNC)*, 162–167. <https://doi.org/10.1109/ICNC64010.2025.10994157>
- Dinh, L., Quang, P. T. A., & Leguay, J. (2024). Towards safe load balancing based on control barrier functions and deep reinforcement learning. *NOMS 2024 – IEEE Network Operations and Management Symposium*, 1–9. <https://doi.org/10.1109/NOMS59830.2024.10575399>

- Fawaz, H., Houidi, O., Zeghlache, D., Lesca, J., Quang, P. T. A., Leguay, J., & Medagliani, P. (2023). Graph convolutional reinforcement learning for load balancing and smart queuing. *2023 IFIP Networking Conference (IFIP Networking)*, 1–9. <https://doi.org/10.23919/IFIPNetworking57963.2023.10186430>
- Feng, L., Xie, R., Tang, Q., & Huang, T. (2024). Delay-prioritized task scheduling with load balancing in computing power networks. *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, 1–6. <https://doi.org/10.1109/WCNC57260.2024.10570622>
- Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 88, 50–71. <https://doi.org/10.1016/J.JNCA.2017.04.007>
- Ghosh, S., Seshadri, K., & Kollengode, C. (2024). Design and evaluation of reward functions for load balancing in cloud datacenters. *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 1–6. <https://doi.org/10.1109/ICITIIT61487.2024.10580611>
- Gu, Y., Cheng, Y., Chen, C. L. P., & Wang, X. (2022). Proximal policy optimization with policy feedback. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(7), 4600–4610. <https://doi.org/10.1109/TSMC.2021.3098451>
- Gupta, M., Chinchali, S., Varkey, P. P., & Andrews, J. G. (2024). Forecaster-aided user association and load balancing in multi-band mobile networks. *IEEE Transactions on Wireless Communications*, 23(5), 5157–5171. <https://doi.org/10.1109/TWC.2023.3324685>
- Han, S., Lee, E., & Pack, S. (2023). Fine-grained load balancing with multi-agent reinforcement learning for self-organizing networks. *2023 IEEE Globecom Workshops (GC Wkshps)*, 578–583. <https://doi.org/10.1109/GCWkshps58843.2023.10464981>
- Hartman, L., Gomez-Rosero, S., Adjei, P., & Capretz, M. A. M. (2023). Multi-factor edge-weighting with reinforcement learning for load balancing of electric vehicle charging stations. *2023 International Conference on Machine Learning and Applications (ICMLA)*, 580–587. <https://doi.org/10.1109/ICMLA58977.2023.00086>
- He, C., Jiang, W., Wang, X., Wang, W., & Xie, X. (2025). Generative AI-enhanced task off-loading strategy for the IoV: An RSU-RSU load-balancing perspective. *IEEE Networking Letters*, 7(3), 161–165. <https://doi.org/10.1109/LNET.2025.3542094>
- Hong, Y., No, J., & Kim, S. (2006). DNS-based load balancing in distributed web-server systems. *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*. <https://doi.org/10.1109/SEUS-WCCIA.2006.23>
- Houidi, O., Bakri, S., Zeghlache, D., Lesca, J., Quang, P. T. A., Leguay, J., & Medagliani, P. (2023). AMAC: Attention-based multi-agent cooperation for smart load balancing. *NOMS 2023 – IEEE/IFIP Network Operations and Management Symposium*, 1–7. <https://doi.org/10.1109/NOMS56928.2023.10154214>

- Huang, C., Wang, F., & Xu, W. (2024). Joint UAV movement control and load balancing based on indirect control in air-ground-integrated networks. *IEEE Wireless Communications Letters*, 13(6), 1616–1620. <https://doi.org/10.1109/LWC.2024.3383858>
- Jeong, Y., Lim, J., Choi, G., & Roh, B. (2024). Deep deterministic policy gradient-based load balancing method in SDN environments. *2024 International Conference on Smart Applications, Communications and Networking (SmartNets)*, 1–6. <https://doi.org/10.1109/SmartNets61466.2024.10577651>
- Jiang, D., Zhu, B., Liu, X., & Mumtaz, S. (2024). ALCoD: An adaptive load-aware approach to load balancing for containers in IoT edge computing. *IEEE Internet of Things Journal*, 11(23), 37480–37492. <https://doi.org/10.1109/JIOT.2024.3427642>
- Jiang, G., Wei, W., Wang, K., Pang, C., & Liu, Y. (2023). DRL-TAL: Deep reinforcement learning-based traffic-aware load balancing in data center networks. *GLOBECOM 2023 – IEEE Global Communications Conference*, 928–933. <https://doi.org/10.1109/GLOBECOM54140.2023.10437481>
- Johann, T., Kuehl, S., & Pachnicke, S. (2024). Deep reinforcement learning based decentralized routing and load-balancing in meshed QKD-networks. *ECOC 2024; 50th European Conference on Optical Communication*, 1385–1388. <https://ieeexplore.ieee.org/document/10926232>
- Jurše, J., & Kuhar, U. (2024). Enhancing low voltage network capacity through phase load balance optimization using advanced smart meter data analytics and static reconnections. *Cired 2024 Vienna Workshop, 2024*, 409–412. <https://doi.org/10.1049/icp.2024.2062>
- Khoramnejad, F., & Hossain, E. (2025). Carrier aggregation, load balancing, and backhauling in non-terrestrial networks: Generative diffusion model-based optimization. *IEEE Transactions on Wireless Communications*, 24(5), 4483–4499. <https://doi.org/10.1109/TWC.2025.3542126>
- Kołakowski, R., Kukliński, S., & Tomaszewski, L. (2024). Hierarchical deep reinforcement learning-based load balancing algorithm for multi-domain software-defined networks. *2024 IFIP Networking Conference (IFIP Networking)*, 607–612. <https://doi.org/10.23919/IFIPNetworking62109.2024.10619899>
- Konar, A., Wu, D., Xu, Y. T., Jang, S., Liu, S., & Dudek, G. (2023). Communication load balancing via efficient inverse reinforcement learning. *ICC 2023 – IEEE International Conference on Communications*, 472–478. <https://doi.org/10.1109/ICC45041.2023.10279136>
- Kopparapu, C. (2002). *Load balancing servers, firewalls, and caches*. Wiley.
- Krishna, M. S. R., & Khasim Vali, D. (2025). Meta-RHDC: Meta reinforcement learning driven hybrid Lyrebird Falcon optimization for dynamic load balancing in cloud computing. *IEEE Access*, 13, 36550–36574. <https://doi.org/10.1109/ACCESS.2025.3544775>
- Kumari, A., Roy, A., & Singh Sairam, A. (2024). Optimizing SDN controller load balancing using online reinforcement learning. *IEEE Access*, 12, 131591–131604. <https://doi.org/10.1109/ACCESS.2024.3459952>

- Lahande, P. V., Kaveri, P. R., Saini, J. R., Kotecha, K., & Alfarhood, S. (2023). Reinforcement learning approach for optimizing cloud resource utilization with load balancing. *IEEE Access*, 11, 127567–127577. <https://doi.org/10.1109/ACCESS.2023.3329557>
- Li, T., Ying, S., Zhao, Y., & Shang, J. (2024). Batch jobs load balancing scheduling in cloud computing using distributional reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems*, 35(1), 169–185. <https://doi.org/10.1109/TPDS.2023.3334519>
- Liu, J., Guo, Y., Leng, X., & Tang, X. (2025). Personalized federated management and load balancing for multiple charging stations. *IEEE Transactions on Industrial Informatics*, 21(8), 6262–6273. <https://doi.org/10.1109/TII.2025.3563534>
- Liu, J., Liu, Z., & Tang, X. (2023). A deep reinforcement learning method for charging station management and load balancing. *2023 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*, 1–5. <https://doi.org/10.1109/ITECAsia-Pacific59272.2023.10372308>
- Liu, L., Chen, Y., Shao, X., Wang, K., & Huang, Y. (2024). Adaptive load balancing strategy for VR microservices with edge AI. *2024 7th International Conference on Electronics Technology (ICET)*, 1064–1069. <https://doi.org/10.1109/ICET61945.2024.10672636>
- Liu, Q., Li, X., Ji, H., & Zhang, H. (2023). User grouping-based beam handover scheme with load-balancing for LEO satellite networks. *GLOBECOM 2023 – IEEE Global Communications Conference*, 3965–3970. <https://doi.org/10.1109/GLOBECOM54140.2023.10436950>
- Ma, J., Liua, X., Hu, H. X. D., Shi, G., & Ma, L. (2025). HydraChain: A cooperative MAPPO architecture for load balancing in IoT sharding blockchain. *IEEE Internet of Things Journal*, 12(15). <https://doi.org/10.1109/JIOT.2025.3567146>
- Mary, A. H., Miry, A. H., & Miry, M. H. (2022). ANFIS based reinforcement learning strategy for control a nonlinear coupled tanks system. *Journal of Electrical Engineering and Technology*, 17(3), 1921–1929. <https://doi.org/10.1007/S42835-021-00753-1/METRICS>
- Mosalli, H., Sanami, S., Yang, Y., Yeh, H., & Aghdam, A. G. (2025). Dynamic load balancing for EV charging stations using reinforcement learning and demand prediction. *2025 IEEE International systems Conference (SysCon)*, 1–7. <https://doi.org/10.1109/SysCon64521.2025.11014850>
- Nakagawa, S., & Mizutani, K. (2023). An implementation of intelligent image analysis load balancing for virtualized environment. *2023 Fourteenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, 1–2. <https://doi.org/10.23919/ICMU58504.2023.10412162>
- Naman, H. A., & Ameen, Z. J. M. (2022). A new method in feature selection based on deep reinforcement learning in domain adaptation. *Iraqi Journal of Science*, 63(2), 817–829. <https://doi.org/10.24996/IJS.2022.63.2.35>
- Nimmala, S., Ramchander, M., Mahendar, M., Manasa, P., Bhavani, D. D., & Raghavendar, K. (2024). Dynamic RL-ACO: Reinforcement learning-based ant colony optimization for load balancing in cloud networks. *2024 5th International Conference on Smart Electronics*

- and Communication (ICOSEC), 475–480. <https://doi.org/10.1109/ICOSEC61587.2024.10722410>
- Ris-Ala, R. (2023). *Fundamentals of reinforcement learning*. Springer Nature. <https://link.springer.com/book/10.1007/978-3-031-37345-9>
- Sanghi, N. (2021). *Deep reinforcement learning with Python: With PyTorch, TensorFlow and Open-AI Gym*. Springer Nature. <https://doi.org/10.1007/978-1-4842-6809-4>
- Santos, J., Wauters, T., Turck, F. D., & Steenkiste, P. (2024). Towards optimal load balancing in multi-zone Kubernetes clusters via reinforcement learning. *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, 1–9. <https://doi.org/10.1109/ICCCN61486.2024.10637606>
- Shahakar, M., Mahajan, S. A., & Patil, L. (2024). ONU: A reinforcement load balancing approach for resource-aware task allocation in distributed systems. *2024 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 1–6. <https://doi.org/10.1109/ESCI59607.2024.10497200>
- Shaikh, M. R. R. (2023). Bayesian network based optimal load balancing in software defined networks. *2023 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 1–5. <https://doi.org/10.1109/ESCI56872.2023.10099730>
- Shang, X., Liu, Z., Gao, D., Yang, D., Zhang, W., Foh, C. H., & Zhang, H. (2025). Computing and network load balancing for decentralized deep federated learning in industrial cyber-physical systems: A multi-task approach. *IEEE Journal on Selected Areas in Communications*, 43(9), 2997–3013. <https://doi.org/10.1109/JSAC.2025.3574618>
- Shi, B., & Chen, F. (2023). A dynamic pricing strategy for load balancing across multiple edge servers. *2023 IEEE International Conference on Web Services (ICWS)*, 329–339. <https://doi.org/10.1109/ICWS60048.2023.00052>
- Shi, Y., Yang, Q., Huang, X., Li, D., & Huang, X. (2023). An SDN-enabled framework for a load-balanced and QoS-aware internet of underwater things. *IEEE Internet of Things Journal*, 10(9), 7824–7834. <https://doi.org/10.1109/JIOT.2022.3231329>
- Sun, A., Yang, F., Wu, W., Wang, T., & Sun, Y. (2023). Deep reinforcement learning-based load balancing algorithm for sliced ultra-dense network. *2023 Eleventh International Conference on Advanced Cloud and Big Data (CBD)*, 27–32. <https://doi.org/10.1109/CBD63341.2023.00014>
- Sun, H., Zhang, H., Ma, H., & Leung, V. C. M. (2025). Joint scheduling, computing, and load balancing for time sensitive traffic in SDN-enabled space-air-ground integrated 6G networks: A federated reinforcement learning approach. *IEEE Transactions on Mobile Computing*, 24(10), 9995–10008. <https://doi.org/10.1109/TMC.2025.3567289>
- Takahashi, H., & Shinomiya, N. (2024). An approximate solution using K-shortest path and reinforcement learning for a load balancing problem in communication networks. *2024 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, 659–660. <https://doi.org/10.1109/ICCE-Taiwan62264.2024.10674659>

- Talpur, A., & Gurusamy, M. (2023). Optimizing vehicle-to-edge mapping with load balancing for attack-resilience in IoV. *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 341–347. <https://doi.org/10.1109/CCNC51644.2023.10060632>
- Tian, B., Pan, X., Wang, F., Hu, S., Zhu, L., Xin, X., & Yu, J. (2025). O-RAN-enabled collaborative multi-access edge computing over optical metro and access networks for enhanced load balancing. *Journal of Lightwave Technology*, 43(14), 6515–6532. <https://doi.org/10.1109/JLT.2025.3566334>
- Tian, S., Xiang, S., Zhou, Z., Dai, H., Yu, E., & Deng, Q. (2025). Task offloading and resource allocation based on reinforcement learning and load balancing in vehicular networking. *IEEE Transactions on Consumer Electronics*, 71(1), 2217–2230. <https://doi.org/10.1109/TCE.2025.3542133>
- Tian, Y., Li, J., Wu, D., Jenkin, M., Jang, S., Liu, X., & Dudek, G. (2023). Policy reuse for communication load balancing in unseen traffic scenarios. *ICC 2023 - IEEE International Conference on Communications*, 309–315. <https://doi.org/10.1109/ICC45041.2023.10278806>
- Vaishnavi, S., Maruti, M. S., & Bhargavi, K. (2023). Forward-backward inverse reinforcement learning for load balancing in SDN. *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*, 01–07. <https://doi.org/10.1109/NMITCON58196.2023.10276272>
- Wang, G., Yang, F., Song, J., & Han, Z. (2025). Resource allocation and load balancing for beam hopping scheduling in satellite-terrestrial communications: A cooperative satellite approach. *IEEE Transactions on Wireless Communications*, 24(2), 1339–1354. <https://doi.org/10.1109/TWC.2024.3508741>
- Wibowo, B., Haq, A., & Zein, M. T. A. A. (2023). A comparative study of load balancing methods in cloud-based village information systems. *2023 IEEE 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 285–289. <https://doi.org/10.1109/ICITISEE58992.2023.10404194>
- Wu, C., & Yang, T. (2024). Load balancing in fog networks using digital twins. *IET International Conference on Engineering Technologies and Applications (ICETA 2024)*, 2024, 98–99. <https://doi.org/10.1049/icp.2024.4194>
- Wu, D., Xu, Y. T., Li, J., Jenkin, M., Hossain, E., Jang, S., Xin, Y., Zhang, C., Liu, X., & Dudek, G. (2023). Learning to adapt: Communication load balancing via adaptive deep reinforcement learning. *GLOBECOM 2023 – IEEE Global Communications Conference*, 2973–2978. <https://doi.org/10.1109/GLOBECOM54140.2023.10437528>
- Wu, G., Wang, H., Zhang, H., Shen, Y., Shen, S., & Yu, S. (2024). Mean-field game-based task-offloaded load balance for industrial mobile edge computing systems using software-defined networking. *IEEE Transactions on Mobile Computing*, 23(12), 13773–13786. <https://doi.org/10.1109/TMC.2024.3437761>
- Wu, Y., Zhang, R., Huang, J., Guo, L., & Wu, J. (2025). Incentive-based two-level scheduling algorithms for load balance in vehicular edge computing. *IEEE Internet of Things Journal*, 12(17), 35497–35509. <https://doi.org/10.1109/JIOT.2025.3579039>

- Xie, R., Feng, L., Tang, Q., Huang, T., Xiong, Z., Chen, T., & Zhang, R. (2024). Delay-prioritized and reliable task scheduling with long-term load balancing in computing power networks. *IEEE Transactions on Services Computing*, 17(6), 3359–3372. <https://doi.org/10.1109/TSC.2024.3495500>
- Xu, J., Guo, H., Shen, H., Raj, M., Wurster, S. W., & Peterka, T. (2023). Reinforcement learning for load-balanced parallel particle tracing. *IEEE Transactions on Visualization and Computer Graphics*, 29(6), 3052–3066. <https://doi.org/10.1109/TVCG.2022.3148745>
- Xu, X., & Zou, T. (2024). Research on flow load balance scheduling strategy in SDN based on reinforcement learning. *2024 Sixth International Conference on Next Generation Data-driven Networks (NGDN)*, 409–413. <https://doi.org/10.1109/NGDN61651.2024.10744167>
- Ye, M., Hu, Y., Zhang, J., Guo, Z., & Chao, H. J. (2023). Reinforcement learning-based traffic engineering for QoS provisioning and load balancing. *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*, 1–10. <https://doi.org/10.1109/IWQoS57198.2023.10188762>
- Yucel, S. (2023). Reinforcement learning approach to server selection and load balancing for collaborative virtual services. *2023 Congress in Computer Science, Computer Engineering & Applied Computing (CSCE)*, 1206–1213. <https://doi.org/10.1109/CSCE60160.2023.00202>
- Zervopoulos, A., Campos, L. M., & Oikonomou, K. (2023). Q-delegation: VNF load balancing through reinforcement learning-based packet delegation. *2023 8th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, 1–6. <https://doi.org/10.1109/SEEDA-CECNSM61561.2023.10470625>
- Zhao, R., Cai, J., Luo, J., Gao, J., & Ran, Y. (2025). Demand-aware beam hopping and power allocation for load balancing in digital twin empowered LEO satellite networks. *IEEE Transactions on Wireless Communications*, 24(6), 5084–5098. <https://doi.org/10.1109/TWC.2025.3545745>
- Zhu, R., Li, G., Zhang, Y., Fang, Z., & Wang, J. (2023). Load-balanced virtual network embedding based on deep reinforcement learning for 6G regional satellite networks. *IEEE Transactions on Vehicular Technology*, 72(11), 14631–14644. <https://doi.org/10.1109/TVT.2023.3279625>