Appendix for "Test Case Prioritization: A Systematic Review using Snowballing and TCPFramework with Approach Combinators"

Abstract

This document includes the online Appendix (available at `https://madeyski.e-informatyka.pl/download/ChojnackiMadeyski25Appendix.pdf`) for a paper entitled "Test Case Prioritization: A Systematic Review using Snowballing and TCPFramework with Approach Combinators" by Tomasz Chojnacki and Lech Madeyski. It consists of three sections: 1) Systematic review protocol, 2) Research reproduction instructions, and 3) Considered evaluation baselines.

1. Systematic review protocol

The following review protocol describes the methods used to carry out a systematic review (SR) using snowballing on the topic of test case prioritization (TCP). The protocol is structured according to the guidelines provided by Kitchenham et al. [1] (the simplified version recommended for Ph.D. students) and the PRISMA-P statement [2] (as recommended by SEGRESS [3]).

Introduction

The purpose of the study is to find relevant TCP datasets and algorithms (with a slight emphasis on machine learning solutions; however, other works should also be included). The reasons for conducting the review are:

- To locate opportunities for further improvement in existing studies.
- To summarize the foundational studies and the current state-of-the-art of TCP research for use in the related work description of the thesis.

Based on the research purpose described above, the following SR research questions (SRRQs) were formulated:

- SRRQ1: *What are the available datasets and subject programs for TCP?*
- SRRQ2: *What are the state-of-the-art TCP algorithms?*

Search process

The search process follows the guidelines by Kitchenham et al. [1] and especially the SEGRESS guidelines [3]. Furthermore, a slightly altered version of the snowballing method proposed by Wohlin [4] and later acknowledged by Kitchenham et al. in the SEGRESS guidelines [3] is used. The requirements and guidelines for different stages of the search process are listed below:

Initial papers: The snowballing approach requires a *start set* of papers to begin the iterative process. Initially, a list of four papers, provided by the second author in the original submission of the thesis topic, is available, namely the works of Yang et al. [5], Marijan [6], Yaraghi et al. [7] and Bagherzadeh et al. [8]. Since these papers were unmethodically cherry-picked, they cannot be directly used as the start set. However, since they are all recent and unrelated publications on the process of TCP, they can provide insight into which terms and keywords are used in the field.

Start set: The extracted list of keywords along with the SRRQs and eligibility criteria described below will be used to construct a search query for the Scopus[1] database. From the search results, a start set of a few unrelated papers published in the last 5 years will be selected. Original submission papers may be included in the start set if deemed appropriate.

Snowballing: Subsequently, for backward snowballing, the reference list located in the paper text should be traversed, and for forward snowballing, Google Scholar[2] is used. Each encountered paper is validated regarding the inclusion and exclusion criteria and gets included in the list if it fulfills all requirements.

Author search: The snowballing guidelines recommend contacting the authors of the articles after the iterative process is concluded [4]. However, this is infeasible to perform in the time allocated for master's thesis completion. Thus, instead of this procedure, if certain authors appear at least five times in the paper collection, their list of publications will be analyzed separately (through Scopus, Google Scholar, ORCID[3], and personal websites) in search

[1]`https://www.scopus.com/search/form.uri`
[2]`https://scholar.google.com`
[3]`https://orcid.org`

of omitted works, without direct contact.

Guidelines by Kitchenham et al. [1] suggest that in the case of a single researcher, techniques used to reduce bias should be noted. To mitigate the issue, all stages of the SR will be presented to the thesis supervisor.

Inclusion and exclusion criteria

According to Wohlin's guidelines [4], the screening is done by examining the title and abstract, the place of the citation, and only if these are insufficient, the full text of the article. All articles should be analyzed at least to the level of abstract inspection, which means that a given article should not be included or excluded solely on the basis of title.

Articles must meet at least one of the following **inclusion criteria** to be included:

1. The article presents a new TCP dataset (SRRQ1).
2. The article describes a new TCP algorithm (SRRQ2).
3. The article is a secondary study on the topic of TCP.
4. The article compares available TCP solutions.

The *paper kind* is defined for included works. A paper has a kind of `SRRQ1` if it fulfills the first condition in the list, `SRRQ2` if it fulfills the second, `SR` if it fulfills the third, and `CMP` if it fulfills the last. A given paper can have multiple kinds (for example, if it introduces both a dataset and an algorithm), and it must belong to at least one kind.

For the purpose of the review, the use of an open-source program as a dataset does not grant the article the kind of `SRRQ2`. However, creating an entirely new dataset, either from existing data or synthetically, modifying an existing dataset, or gathering own data internally, all count towards this label.

Articles that violate any of the following **exclusion criteria** will be excluded:

1. The paper places the main emphasis on test case generation, selection, or minimization.
2. The paper is unrelated to the Computer Science research area.
3. The paper was not peer-reviewed or has not yet been published.
4. The paper consists of less than 5 pages.
5. The paper was not written in the English language.

The third exclusion criterion implicitly disallows the inclusion of blogs, vlogs, or other types of gray literature. Although engineering blogs may provide interesting solutions, they are often available in the form of a research paper or are biased.

Furthermore, while not an explicitly formulated exclusion criterion, papers published after January 27th, 2025 (the commencement date of the SR search process) will not be included. These articles would not have been available during the search.

Data collection

The following information will be collected for each article:

- Digital Object Identifier (DOI) of the paper, if available.
- Title of the paper.
- Authors of the article.
- Year of publication (the lowest date if it was published multiple times).
- Search stage (e.g., *Iteration #1: Backward*) in which the paper was found.
- Paper kind of the article (as described in the previous section).
- Short notes regarding the paper (mainly names of datasets or algorithms).

It is sometimes recommended to maintain the list of excluded papers, similarly to the list of inclusions [1]. However, because of the huge amount of papers evaluated during snowballing, some of which are obviously unrelated to the studied topic, it was decided not to keep a list of excluded papers. Individual excluded papers might get highlighted if the circumstances of their exclusion are particularly interesting.

Data analysis

The resulting list of included papers alongside the collected data, ordered by the inclusion order, will be kept. All records will be stored in a spreadsheet file using the XLSX format and published in the reproduction package.

The following aggregations of results will be analyzed:

- total number of new papers included in each search process stage,
- total number of papers per year,
- total number of papers per kind.

All identified SRs covering the entire research area will be listed chronologically, along with relevant information: target years, limitations, and analyzed paper counts. Although other SRs are of little interest in the context of this study, they greatly aid in the snowballing process, since they include many references.

The most relevant papers, namely those that describe foundational aspects of TCP, introduce new datasets, or produce the state-of-the-art results, will be further analyzed as part of the thesis research. Different subareas of the topic will be summarized, as well as key definitions and metrics.

The results of the study will be published as part of the author's master's thesis. The detailed search process advances along with any divergences from the protocol will be described in the relevant chapter of the main work.

2. Research reproduction instructions

The following appendix describes the research reproduction instructions, including the project architecture, required dependencies, useful commands for replicating the research questions, and, finally, guidelines for the extension of TCPFramework. The source code can be accessed at `https://github.com/LechMadeyski/MSc25TomaszChojnacki`.

Running the code

The officially supported way to set up the repository is to use uv[4] – a popular Python package and project manager written in Rust. The uv executable can be installed through standalone installers available for download from the project website or through pip.

To run the project as is, selected repositories of the RTPTorrent [9] dataset are required, which are not included in the replication package. Firstly, the ZIP file from the RTPTorrent Zenodo package[5] should be downloaded. Inside, the `rtp-torrent` directory contains subfolders for different

[4]`https://docs.astral.sh/uv`
[5]`https://zenodo.org/records/4046180`

source programs. The main CSV file from each project directory should be extracted into the `tcp-framework/datasets` directory and optionally renamed. Additionally, the `tr_all_built_commits.csv` file should be moved to the same directory (alternatively, the `travistorrent_8_2_2017.csv` from the TravisTorrent [10] dataset can be used to achieve more reliable applicability results). Finally, the Git repositories for the selected projects should be cloned into the `datasets` directory, under the same names as their corresponding CSV files. Hyperlinks to the repositories used in this study are presented in Table 1.

Table 1: Hyperlinks to the project repositories used in the study.

System	**Repository**
`LittleProxy`	`github.com/adamfisk/LittleProxy`
`HikariCP`	`github.com/brettwooldridge/HikariCP`
`jade4j`	`github.com/neuland/jade4j`
`wicket-bootstrap`	`github.com/martin-g/wicket-bootstrap`
`titan`	`github.com/thinkaurelius/titan`
`dynjs`	`github.com/dynjs/dynjs`
`jsprit`	`github.com/graphhopper/jsprit`
`DSpace`	`github.com/DSpace/DSpace`
`optiq`	`github.com/julianhyde/optiq`
`cloudify`	`github.com/CloudifySource/cloudify`
`okhttp`	`github.com/square/okhttp`

Scripts used for research questions can be evaluated using the `uv run` subcommand, that is, `uv run rq2.py`, `uv run rq3.py`, and other scripts using TCPFramework can be executed analogously.

Extending the project

The solution is highly expandable. New approaches should be implemented as subclasses of the `Approach` abstract base class. For simple prioritizers, only the `prioritize` method has to be implemented. It should call the `ctx.execute` function on subsequent test cases extracted from `ctx.test_-cases` property. More advanced techniques may optionally override the `get_dry_ordering`, `on_static_feedback`, and `reset` methods. Their usage examples can be seen by inspecting the approaches packed in the `approaches` module.

Custom datasets can be included by subclassing the `Dataset` class. Most importantly, the `cycles` method should be overriden. The `MetricCalc` class is not designed with expandability in mind; the easiest way to calculate new metrics is to modify its code. On the other hand, the approaches related to code representation and approach combinators are highly modular and should be easy to extend.

3. Considered evaluation baselines

Two major steps were used to find more comparison points: we checked the most popular baselines for TCP evaluation from Prado Lima and Vergilio [11], and collected all prioritization approaches of the previous five years found in the SR. A total of 60 techniques were reviewed, 54 were excluded for various reasons, and there were 3 duplicates. In the end, three new techniques were included as baselines, ROCKET by Marijan et al. [12], DBP by Zhou et al. [13], and an unnamed approach by Fazlalizadeh et al. [14]. There were multiple exclusion reasons, as outlined below:

- *Incompatible*: Some methods proposed by other researchers use incompatible problem statements, making them uncomparable with our techniques. This mostly means that they use various resource *constraints* (e.g. seeing how the suite behaves when only 50% of testing time budget is used), require *manual* work (e.g. using expert insights to prioritize), or use the fault *matrix* instead of real CI data.

- *Awareness*: Other techniques may use more information factors than our methods. Some of these are not available in RTPTorrent (*requirement* association, code *model*s, *bug* reports, emitted *log*s or *UI* elements). Others (like *change* and *coverage* information) are derivable, but were not used by any of our techniques. We therefore exclude them to ensure fair comparison.

- *Training*: A major upside of approach combinators is that they do not require any prior training or fine-tuning. Since they work from the first CI cycle inclusively, it would be unfair to compare them with approaches applicable only after some cycles passed. We also include

ROCKET, which was shown to be better than many ML methods Marijan [6], further reducing the risks caused by the absence of ML approaches among the baselines.

Figure 1 shows the counts of entries included or excluded for various reasons, and Table 2 shows the detailed outline of all the baseline approaches considered.

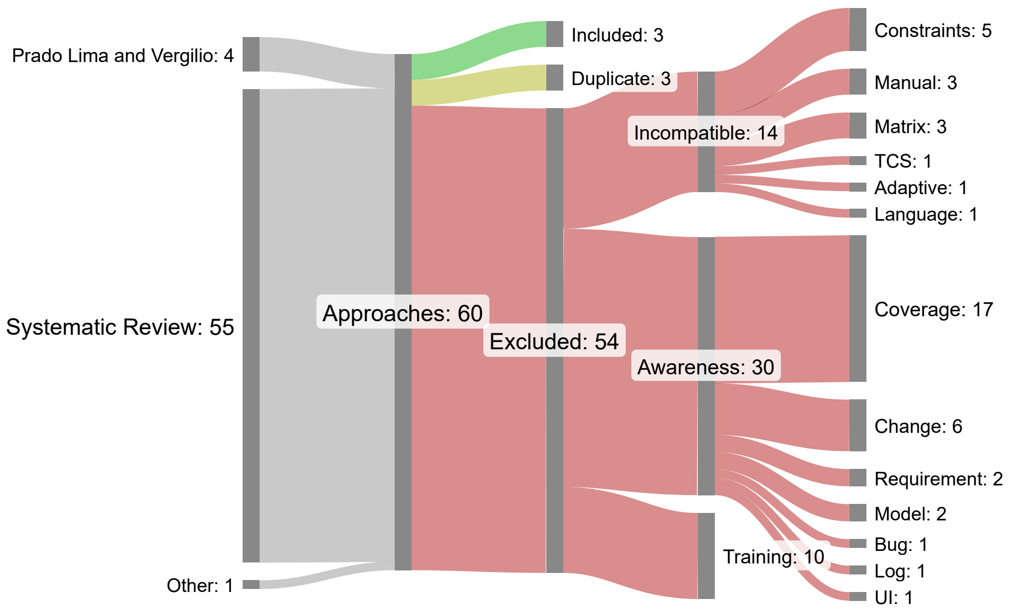

Figure 1: A Sankey diagram of counts of considered baselines.

Table 2: Index of all considered evaluation baselines.

#	Approach (DOI)	Verdict	Reason
1	random (`10.1109/ICSM.1999.792604`)	–	duplicate
2	ROCKET (`10.1109/ICSM.2013.91`)	✓	
3	no prioritization (`10.1109/ICSM.1999.792604`)	–	duplicate
4	manual (`N/A`)	✗	incompatible → manual
5	SSTcp (`10.1109/ISSRE59848.2023.00077`)	✗	awareness → change
6	CCCP (`10.1016/j.jss.2020.110712`)	✗	awareness → coverage
7	COLEMAN (`10.1109/TSE.2020.2992428`)	✗	incompatible → constraints
8	NEUTRON (`10.1109/ASEW60602.2023.00014`)	✗	incompatible → constraints
9	CovClustering (`10.1007/s10515-022-00344-y`)	✗	awareness → coverage
10	TCP-NET (`10.1109/ICSTW55395.2022.00034`)	✗	awareness → coverage
11	TFC-SVM (`10.1007/978-981-19-0976-4_20`)	✗	training
12	H1/H2/H3 (`10.1007/s00500-022-07236-z`)	✗	awareness → requirement
13	LoM-Score (`10.1109/ACCESS.2023.3283212`)	✗	awareness → coverage
14	Genetic-Diff (`10.1109/SEAA56994.2022.00028`)	✗	awareness → coverage
15	Contextual (`10.1016/j.infsof.2024.107444`)	✗	incompatible → constraints
16	EVDMOA (`10.1016/j.eswa.2025.126634`)	✗	awareness → coverage
17	TransBoost (`10.1109/AST58925.2023.00023`)	✗	training
18	RETECS (`10.1109/ACCESS.2021.3063232`)	✗	incompatible → constraints
19	TK RTS (`10.1002/smr.2653`)	✗	awareness → change
20	SBLA-AB-CNN (`10.1007/s11227-022-04540-1`)	✗	incompatible → TCS
21	*unknown* (`10.1007/978-981-19-3148-2_34`)	✗	awareness → change
22	DeepRP (`10.22266/ijies2024.0229.64`)	✗	training
23	RETECS (`10.1109/QRS57517.2022.00088`)	✗	incompatible → constraints
24	GAIL (`10.1109/AINIT61980.2024.10581812`)	✗	training
25	DLTCP (`10.1016/j.eswa.2024.124768`)	✗	awareness → model
26	CAP (`10.1016/j.infsof.2023.107339`)	✗	incompatible → adaptive
27	OCP (`10.1016/j.jss.2022.111419`)	✗	awareness → coverage
28	PLI-based EDAS (`10.3390/math8111857`)	✗	incompatible → manual
29	OTCP (`10.32604/cmc.2021.014686`)	✗	training
30	MWDM-IGA (`N/A`)	✗	awareness → coverage
31	*unknown* (`10.1007/978-3-031-80889-0_4`)	✗	awareness → coverage
32	1-D (`10.14569/IJACSA.2023.0140698`)	✗	awareness → coverage
33	QAC-TCP (`N/A`)	✗	awareness → coverage
34	MRFR/EDR (`10.21123/bsj.2024.9604`)	–	duplicate
35	C-CORE (`10.32604/cmes.2023.043248`)	✗	incompatible → language
36	MOTCP (`10.2478/jee-2024-0018`)	✗	incompatible → manual
37	RTCP (`10.1109/ASSIC60049.2024.10507913`)	✗	awareness → requirement
38	*unknown* (`10.30534/ijeter/2020/104852020`)	✗	incompatible → matrix
39	APFDNet (`10.2139/ssrn.4866654`)	✗	training
40	DBP (`10.1109/TR.2020.2979815`)	✓	
41	iBAT (`10.1007/s00500-022-07121-9`)	✗	incompatible → matrix
42	*unknown* (`10.1007/s00500-020-05517-z`)	✗	awareness → bug

43	LogTCP (`10.1145/3569932`)	✗	awareness → log
44	AGA (`10.1109/TSE.2021.3137929`)	✗	awareness → coverage
45	DeepOrder (`10.1109/ICSME52107.2021.00053`)	✗	training
46	AFSA (`10.1016/j.comcom.2021.09.014`)	✗	awareness → coverage
47	Colosseum (`10.1109/TSE.2021.3111169`)	✗	awareness → change
48	SegTCP (`10.1145/3650212.3680349`)	✗	awareness → UI
49	TCP-Tune (`10.1109/ICSTW60967.2024.00014`)	✗	awareness → change
50	*unknown* (`10.1145/3526072.3527525`)	✗	training
51	*unknown* (`10.1109/APSEC53868.2021.00075`)	✗	training
52	*unknown* (`10.1007/978-981-99-6706-3_7`)	✗	awareness → coverage
53	*unknown* (`10.1109/ICSTW52544.2021.00035`)	✗	awareness → change
54	ABO (`10.32604/cmc.2023.032308`)	✗	incompatible → matrix
55	LTR-TS (`10.1109/ACCESS.2021.3053163`)	✗	awareness → model
56	SFLA (`10.32604/cmc.2023.031261`)	✗	awareness → coverage
57	HSP (`10.1016/j.jss.2019.110430`)	✗	awareness → coverage
58	*unknown* (`10.1145/3644032.3644467`)	✗	training
59	REM-WOA (`10.15837/ijccc.2023.2.5049`)	✗	awareness → coverage
60	*unknown* (`10.1007/978-3-642-02949-3_5`)	✓	

References

[1] B. Kitchenham, S. Charters, D. Budgen, P. Brereton, M. Turner, S. Linkman, M. Jørgensen, E. Mendes, G. Visaggio, Guidelines for performing systematic literature reviews in software engineering, Technical Report, Keele University, 2007. URL: `https://madeyski.e-informatyka.pl/download/Kitchenham07.pdf`, version 2.3, retrieved February 22, 2025.

[2] D. Moher, L. Shamseer, M. Clarke, D. Ghersi, A. Liberati, M. Petticrew, P. Shekelle, L. A. Stewart, PRISMA-P Group, Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement, Systematic Reviews 4 (2015) 1. doi:`10.1186/2046-4053-4-1`.

[3] B. Kitchenham, L. Madeyski, D. Budgen, SEGRESS: Software Engineering Guidelines for REporting Secondary Studies, IEEE Transactions on Software Engineering 49 (2023) 1273–1298. doi:`10.1109/TSE.2022.3174092`.

[4] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th

International Conference on Evaluation and Assessment in Software Engineering, EASE '14, Association for Computing Machinery, 2014, pp. 1–10. doi:10.1145/2601248.2601268.

[5] L. Yang, J. Chen, H. You, J. Han, J. Jiang, Z. Sun, X. Lin, F. Liang, Y. Kang, Can code representation boost IR-based test case prioritization?, in: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), 2023, pp. 240–251. doi:10.1109/ISSRE59848.2023.00077.

[6] D. Marijan, Comparative study of machine learning test case prioritization for continuous integration testing, Software Quality Journal 31 (2023) 1415–1438. doi:10.1007/s11219-023-09646-0.

[7] A. S. Yaraghi, M. Bagherzadeh, N. Kahani, L. C. Briand, Scalable and accurate test case prioritization in continuous integration contexts, IEEE Transactions on Software Engineering 49 (2023) 1615–1639. doi:10.1109/TSE.2022.3184842.

[8] M. Bagherzadeh, N. Kahani, L. Briand, Reinforcement learning for test case prioritization, IEEE Transactions on Software Engineering 48 (2022) 2836–2856. doi:10.1109/TSE.2021.3070549.

[9] T. Mattis, P. Rein, F. Dürsch, R. Hirschfeld, RTPTorrent: An open-source dataset for evaluating regression test prioritization, in: Proceedings of the 17th International Conference on Mining Software Repositories, MSR '20, Association for Computing Machinery, 2020, p. 385–396. doi:10.1145/3379597.3387458.

[10] M. Beller, G. Gousios, A. Zaidman, TravisTorrent: Synthesizing Travis CI and GitHub for full-stack research on continuous integration, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), 2017, pp. 447–450. doi:10.1109/MSR.2017.24.

[11] J. A. Prado Lima, S. R. Vergilio, Test case prioritization in continuous integration environments: A systematic mapping study, Information and Software Technology 121 (2020) 106268. doi:10.1016/j.infsof.2020.106268.

[12] D. Marijan, A. Gotlieb, S. Sen, Test case prioritization for continuous regression testing: An industrial case study, in: 2013 IEEE International

Conference on Software Maintenance, 2013, pp. 540–543. doi:10.1109/ICSM.2013.91.

[13] Z. Q. Zhou, C. Liu, T. Y. Chen, T. H. Tse, W. Susilo, Beating random test case prioritization, IEEE Transactions on Reliability 70 (2021) 654–675. doi:10.1109/TR.2020.2979815.

[14] Y. Fazlalizadeh, A. Khalilian, M. Abdollahi Azgomi, S. Parsa, Incorporating historical test case performance data and resource constraints into test case prioritization, in: C. Dubois (Ed.), Tests and Proofs, Springer Berlin Heidelberg, 2009, pp. 43–57. doi:10.1007/978-3-642-02949-3_5.