Reproducibility Report for ACM SIGMOD 2023 Paper: "Parallel Strong Connectivity Based on Faster Reachability"

SEYED MAHMOUD SAJJADI MOHAMMADABADI, University of Nevada, Reno, USA
REHAM OMAR, Concordia University, Canada

The reproducibility evaluation successfully replicated key functionalities of LightRW, particularly MetaPath and Node2Vec random walk accelerators. Challenges were encountered in modifying the FPGA shell, yet core functionalities showcased the potential of FPGA acceleration in dynamic graph algorithms. The authors provided data files, experiment scripts, and plotting scripts that allowed reproduction of the main results.

1 INTRODUCTION

In the realm of graph theory, strongly connected components (SCCs) are pivotal structures, but their parallel computation on large-scale graphs poses significant challenges. This work delves into the development of a novel parallel SCC algorithm, leveraging innovative techniques in reachability computation and synchronization control.

Addressing the limitations of existing parallel implementations, this approach focuses on enhancing parallelism and mitigating scheduling overhead through vertical granularity control (VGC). Central to this solution is the efficient utilization of parallel hash bags to manage vertices and optimize frontier maintenance during rounds of computation. This paper [1] presents a comparative analysis of the authors' parallel SCC implementation against established systems like GBBS, iSpan, and Multi-step on 18 directed graphs, showcasing substantial performance improvements.

2 SUBMISSION

Leveraging primitives from the public repository GBBS, derived from pbbslib, this system targets multi-processor machines and requires specific software dependencies like g++, Python 3 with essential libraries, matplotlib, seaborn, and more.

This submission includes:

- GitHub repository hosting code and scripts available at: https://github.com/ucrparlay/Parallel-Strong-Connectivity
- Detailed README file outlining implementation details and usage instructions
- Data generators and sources used in the evaluation, accessible at specified URLs (https://github.com/ucrparlay/Parallel-Strong-Connectivity/blob/main/scripts/graphs.py)

3 HARDWARE AND SOFTWARE ENVIRONMENT

The following list outlines the hardware and local machine setup utilized for the experiments conducted in the paper "Parallel Strong Connectivity based on Faster Reachability" presented at SIGMOD '23. The deployment required specific software dependencies and a multi-processor machine configuration as described below.

The experimentation encompassed diverse benchmarks and scalability tests across various algorithms, conducted on a machine featuring:

- CPU: 4x Intel(R) Xeon(R) Gold 6252 CPU @ 2.10GHz
- Physical CPU cores: 96
- Threads per core: 2
- NUMA nodes: 4
- Memory: 1510 GB
- Operating system: CentOS Stream 8

The requirements for deployment involved tools such as g++, Python 3 with specific versions of essential libraries, matplotlib, seaborn, parlaylib for fork-join parallelism, Bazel 2.1.0, and cmake 3.14+. The experiment scripts were designed to download datasets and compile algorithms while providing provisions for scalability testing and data collection.

The script-based experiments aimed to generate data for multiple figures and tables used in the paper. However, the complexity of the environment and dependencies might pose challenges in replicating the exact environment and obtaining consistent results.

4 REPRODUCIBILITY EVALUATION

4.1 Process

The repository aims to replicate algorithms related to strongly connected components, connectivity, and LE-Lists. The deployment required a multi-processor machine and specific software tools, including g++, Python 3 with requisite libraries, parlaylib, Bazel 2.1.0, and cmake 3.14+. The setup was tested on two machines: a machine equipped with multiple physical CPU cores and CentOS operating system, and the author's server.

The reproducibility process involved following documented instructions, including cloning the repository and its submodules. The preparation step included downloading datasets and compiling essential algorithms using provided scripts. The experiments covered 18 benchmarks, generating outputs such as running times, scalability tests, and algorithmic performances on some graphs.

4.2 Results

The reproducibility evaluation successfully replicated some of the outlined environments and procedures mentioned in the documentation. Following the provided instructions, we were able to execute some experiments, such as running algorithms and collecting data. Notably, we were able to reproduce the scalability tests (Figure 7, 8, 11) and running time analyses (Table 3, Figure 1) related to strongly connected components to the extent that our hardware allowed. However, we encountered some challenges in preparing the environment.

Due to the time-consuming nature of experiment-build processes and the complexity of some experiments, achieving comprehensive validation of all functionalities within the evaluation timeframe was not feasible. Nevertheless, we successfully replicated the core functionalities critical to the algorithms proposed in the paper, indicating the potential for reproducibility of key aspects of the experiments.

The results, encompassing tables and figures generated from experiments, aligned with the described outcomes in the paper. Considering the complexity of the environment and its dependencies, however, achieving complete replication of all experiments and obtaining consistent results might require further investigation and validation.

Table 1 shows the results reported in the paper, compared to the reproduced results for the two main settings: parallel and sequential execution. The error percentage column shows the divergence of the reproduced results from the reported results, if the reproduced result is equal to or smaller than the reported results the value is empty since this indicates a better performance. As shown in the table, the error percentage does not exceed 10% which is the error margin indicated by the authors for the experiments.

5 SUMMARY

The submission provides extensive documentation, including a GitHub repository containing code and scripts, a detailed README file with implementation instructions, and accessible data generators utilized for evaluation. The reproducibility evaluation successfully replicated core

Table 1. Comparison between paper and reproducibility values of the first two columns in [1] namely the parallel and sequential execution times

	Parallel			Sequential		
Graph	Paper	Reproducibility	error (%)	Paper	Reproducibility	error (%)
LJ	0.038	0.041	7.9	1.06	0.732	-
TW	0.226	0.225	-	14.3	8.66	-
SD	1.96	1.65	-	104	72.25	-
CW	17.6	16.55	-	1189	1282	7.8
HL14	20.6	20.0	-	1622	1461	-
HL12	95.5	88.8	-	8528	6879	-
HH5	0.208	0.200	-	3.1	2.06	-
CH5	0.557	0.544	-	5.83	4.45	-
GL2	0.598	0.621	3.8	39.1	21.51	-
GL5	0.865	0.899	3.9	45.8	26.67	-
GL10	1.49	1.52	2.0	61.6	39.05	-
GL15	2.09	2.18	4.3	75.5	50.15	-
GL20	2.38	2.33	-	86	59.88	-
COS5	3.22	3.37	4.7	284	140.82	-
SQR	0.577	0.111	-	24.7	2.28	-
REC	0.117	0.113	-	2.08	1.41	-
SQR'	1.38	0.271	-	105	9.17	-
REC'	0.159	0.160	-	9.38	5.29	-

functionalities, despite the complex setup requiring specific OS, kernel versions, FPGA platforms, and modified shells.

REFERENCES

[1] Letong Wang, Xiaojun Dong, Yan Gu, and Yihan Sun. 2023. Parallel Strong Connectivity Based on Faster Reachability. *Proc. ACM Manag. Data* 1, 2 (2023), 114:1–114:29. https://doi.org/10.1145/3589259