Reproducibility and Availability: INEv: In-Network Evaluation for Event Stream Processing

Samira Akili Steven Purtzel
Matthias Weidlich
Humboldt-Universität zu Berlin

In-Network (INEv) evaluation graphs provide a model for the in-network evaluation of Complex Event Processing (CEP) queries. Given a query workload, network topology, and statistics about the event generation in the network, as well as query selectivities, an INEv graph compactly describes (i) how the queries of the workload should be split into sub-queries (query projections), (ii) where the subqueries should be placed in the network, and (iii) how events, matches of projections, and partial results thereof shall be forwarded in the network to reduce network communication.

This document provides a comprehensive guide on reproducing the experimental results of our paper and describes the utilization of INEv for the in-network evaluation of arbitrary workloads. Should you require any assistance with the evaluation or have any inquiries, please do not hesitate to contact us at akilsami@hu-berlin.de.

1 Availability

Our code is available as open-source projection on Git-Hub: https://github.com/samieze/INEv. The repository contains the code for the INEv graph construction and for the generation of input networks and query workloads. While the main contribution of our work is the efficient construction of INEv graphs, we also provide a prototype distributed CEP (DCEP) engine, that enables INEv graph based query evaluation on both synthetic data and real-world datasets. The repository also includes an archive that contains the scripts and data needed to reproduce the experiments from our paper.

2 Reproducibility

We provide full reproducibility of all experiments described in our paper.

Download Materials. Our experiments are categorized in two groups: the *local experiments* and the *pi cluster experiments*. The materials to conduct both series of experiments are in the GitHub repository (https://github.com/samieze/INEv/tree/main/Reproducibility_Submission). The repository contains four archives, two for the local and two for the pi cluster experiments.

There are two options on how to conduct the local experiments:

1. run a single python script that starts all experiments and generates plots for the experiment results
2. pull a Docker container in which the respective python script is run

To conduct the local experiments with Docker, please follow the instructions of the file *README-localexperiments-docker.md*. To run the experiments by running python scripts, download and merge the two archives *local_ experiments.xz.** and follow the instructions detailed out in subsection 2.1. To merge the two archives, execute: `'cat local_experiments.xz.* | unxz | tar x'`.

The total runtime for the local experiments is about **72 hours**.

The second group of experiments have to be executed in a distributed environment. We executed the respective experiments on a proprietary cluster from mythic beasts. To conduct the experiments, download and merge the two archives *pi_ cluster_ experiments.xz.**. To merge the two archives, execute: `'cat pi_cluster_experiments.xz.* | unxz | tar x'`. We provide an extensive instruction on how to provision and configure up a Raspberry Pi cluster with mythic beasts in order to run the experiments as well as scripts that automate the remote execution of experiments, the collection of experiment results and the generation of plots.

2.1 Local experiments

The main part of our experimental results can be reproduced by running the script *./all_figures.sh* within the folder *reproducibility/local_ experiments*. The total runtime of the script should be around **72 hours**. After the scripts termination the folder *reproducibility/local_ experiments/Figures* should contain the figures **Fig.4, Fig.6-Fig.10** as well as **Table 3**.

Hardware Configuration. The system requirements for running the script are:

1. An x86-64 machine running Ubuntu 18.04 or later.
2. Python version 3.8.
3. openjdk 11.0.19.
4. At least 8 logical cores.

We tested the script on a server running Intel Xeon E7-4880 at 2.50GHz with 1TB RAM and 60 logical cores.

Dependencies. The list of used packages is given in *requirements.txt* (inside the folder *reproducibility/local_ experiments*). Please install the required packages, e.g., by running *'pip3 install -r requirements.txt'* from inside the folder *reproducibility/local_ experiments*.

2.1.1 Local Experiment Subfolders.

In this section, we give a brief overview over the different experiments in the *reproducibility/local_ experiments* folder, their expected results and runtime. However, as stated above, all of the local experiments can be executed by evoking the script *./all_ figures.sh*.

REPRO_ basic contains the version of our algorithms as they can be found in the github repository. This version is used to generate INEv graphs under varying network and workload characteristics. Moreover the folder contains the experiments that compare the costs of INEv graphs to a lower bound and investigate the computation time. The following plots are generated by running *./REPRO_ basic/scripts/combined.sh*:

- Fig. 5 (lower bound)
- Fig. 7a-d (Workload Characteristics)
- Fig. 8 (Kleene, Negation)
- Fig. 12 (Computation Time)

The experiment takes about **72 hours** to finish.

REPRO_ adaptivity contains experiments on the robustness of INEv graphs as well as the benefit of the adaptive repair proposed in Section 6.6 of our paper. The following plots are generated by running *./REPRO_ adaptivity/scripts/ combined.sh*.

- Fig. 10a-d (Rate Swaps)
- Fig. 11b (Changed Selectivities)
- Fig. 11c (Changed Edges)

The experiment takes about **24 hours** to finish.

REPRO_ multiquery contains an experiment that investigates the benefit of our approach for multi-query sharing as described in Section 6.3. The following plot is generated by running *./REPRO_ multiquery/scripts/combined.sh*:

- Fig. 9a

The experiment takes about **10 hours** to finish.

REPRO_muse_INEv contains the experiments conducted to compare the transmission costs of MuSE and INEv graphs for networks with varying characteristics (event rate distribution, network size) as part of the State-of-the-art comparison. The following plots are generated by running *./REPRO_muse_Inev /scripts/combined.sh*:

- Fig. 4c-d

The experiment takes about **12 hours** to finish.

REPRO_timewindows contains experiments on the influence of different timewindows for queries of the same workload under varying workload sizes. The following plots are generated by running *./REPRO_timewindows/scripts/ combined.sh*:

- Fig. 9c-d

The experiment takes about **12 hours** to finish.

REPRO_wlorder contains an experiment that investigates the influence of the order in which the queries of a workload are processed by our algorithm. The following plots are generated by running *./REPRO_wlOrder/scripts/ combined.sh*:

- Fig. 9b

The experiment takes about **8 hours** to finish.

REPRO_PPoP_INEv contains the experiments conducted to compare the transmission costs of the distributed evaluation plans proposed by the PPoP approach with INEv graphs for networks with varying event rate distribution. The following plots are generated by running *./REPRO_PPoP_INEv/scipts /combined.sh*:

- Fig. 4a-b

The experiment takes about **8 hours** to finish.

REPRO_variance contains the experiment that investigate the influence of variance in the event rates on the network costs of a given INEv graph. The experiment evaluates 10 INEv graphs with our distributed CEP engine. The 10 INEv graphs used are the same as used for the evaluation of the paper. The following plot is generated by running *./REPRO_variance/scripts/combined.sh*:

- Fig. 11a

The experiment takes about **12 hours** to finish.

REPRO_table. contains all experiments that need to be run in order to obtain Table 3. The experiment evaluates different query workloads on different partitionings of the two real-world data sets google(cluster) and citibike based on distributed evaluation plans obtained by INEv, MuSE, PPoP and a simple hash partitioning scheme inspired by Flink. By running the script *REPRO_table/scipts/combined.sh* a table comprising the results of **Table 3** is generated. The experiment runs for about **24 hours**.

A row in the resulting table yields the transmission ratio for an evaluation plan generated by an approach for a given query and network whereas the network is obtained by partitioning the data set with a given partitioning size. The table schema is given by [Dataset, Query, PartitioningSize, TransmissionRatio, Terminated, Approach]. The field Terminated indicates if the evaluation terminated. If value of Terminated is set to "False", this corresponds to the cells in Table 3 of the paper containing a ">".

Interpretation of Results. All the experiments mentioned above utilize at least one of the following techniques: randomized event rate generation, randomized selectivity generation, randomized query workload generation, or randomized network topology generation. Consequently, the resulting plots may not be identical to those displayed in the paper. However, the trends and relationships observed in the presented data should still support all the key assertions made in the paper.

2.2 Pi Cluster Experiments

The folder *reproducibility/pi_cluster_experiments* contains the subset of our experiments that need to be executed on a real-world distributed environment, a Raspberry Pi Cluster. The folder contains an additional readme file which explains how to provision and prepare a Pi cluster from mythic beasts, run the experiments and generate the respective plots. Following the instructions of the contained *readme.md* file to execute the experiments, the plots generated are: **Fig. 6a-d**. The experiments take about **72 hours** to finish.