Fast Leiden Algorithm for Community Detection in Shared Memory Setting

Subhajit Sahu[1], Kishore Kothapalli[1], and Dip Sankar Banerjee[2]

[1] International Institute of Information Technology Hyderabad, India
{subhajit.sahu@research., kkishore@}iiit.ac.in
[2] Indian Institute of Technology Jodhpur, India
dipsankarb@iitj.ac.in

Getting Started Guide

The source code for our experiments is written in C++. Running our scripts, which involve downloading datasets, conducting experiments, and processing output logs into CSV format, necessitates installations of "**wget**," "**gzip**," "**tar**," "**git**," "**stdbuf**," "**g++**" version 9.4 or higher, **Node.js** version 16 or higher, and **"conda"**. The scripts are named as follows:

- **download-graphs.sh**: Used to download the input graphs for the experiments.
- **run-*.sh**: Used to run the experiments on our Parallel Leiden, i.e., GVE-Leiden.
- **test-*.sh**: Used to run the experiments on existing state-of-the-art implementations.
- **process-*.js**: Used to process the output logs from the experiments into CSV.

Once CSV files are generated for each experiment, they can be imported into the "**data**" sheet of the linked Google Sheets to view updated plots/figures in the "**compare**"/"**all**" sheet (for comparison with figures in our paper). We recommend a system with at least **64 cores** for running the experiments. For testing *cuGraph Leiden*, we recommend a system with **NVIDIA A100 GPU**.

Step-by-Step Instructions

Downloading Large real-world graphs

We use 13 large real-world graphs from the SuiteSparse Matrix Collection. These can be downloaded from the command-line using our provided scripts, as follows:

```
$ bash download-graphs.sh [data-dir]
```

Specifying the data directory is optional, if not provided, the graphs will be downloaded to the "***./Data***" directory. Downloading the graphs may take a long time.

Measuring Performance of our Parallel Leiden (GVE-Leiden)

In **Section 5.2** of our paper, we conduct performance comparisons on large real-world graphs. After downloading the graphs, the experiment can be executed via command-line using the provided scripts, as outlined below:

```
$ bash run-main.sh [data-dir] [logs-dir]
```

Specifying the data and logs directories is optional. If not specified, graphs will be accessed from "**./Data**" and output logs will be saved to "**./Logs**". Running the experiment typically requires approximately one day.

After the experiment concludes, the logs will be stored in "**[logs-dir]/leiden-communities-openmp--main-icpp2024.log**". This log file can be converted to a CSV using the following:

```
$ node process-main.js csv <log-file> <log-file>.csv
```

With the CSV file generated, it can be imported into the "**data**" sheet of the puzzlef/leiden-communities-openmp--icpp spreadsheet. This action will update figures in the "**compare**", **"splits"**, and **"hardness"** sheets, which should resemble **Figures 6, 7 and 8** in our paper.

Measuring Performance of Original Leiden (libleidenalg)

In **Section 5.2** of our paper, we compare the performance of GVE-Leiden with the original Leiden. Running this experiment involves some manual work, in case you are stuck, do contact us. After downloading the graphs, this experiment can be executed via command-line using our provided scripts, as outlined below:

```
$ bash test-leibleidenalg.sh [data-dir] [logs-dir]
```

You can specify the data and logs directories optionally; if not provided, the graphs will be accessed from the "**./Data**" directory, and the output logs will be saved to the "**./Logs**" directory. Expect the experiment to run for approximately 2 days. Once the experiment is complete, the logs will be saved in "***[logs-dir]/vtraag--libleidenalg--main-icpp2024.log***". This log file can be converted to a CSV file as follows:

```
$ node process-leibleidenalg.js csv <log-file> <log-file>.csv
```

With the CSV file generated, import it into a new sheet of the puzzlef/leiden-communities-openmp--icpp spreadsheet. Then transfer these numbers (*total_time* to *vtraag-t*, *modularity* to *vtraag-m*, and *disconnected_communities_fraction* to *vtraag-d*) to the **“compare”** sheet with column names **“vtraag-*”**. This action will update the figures in the "**compare**" sheet, resembling **Figure 6** in our paper.

Measuring Performance of igraph Leiden

In **Section 5.2** of our paper, we compare the performance of GVE-Leiden with igraph Leiden. Running this experiment involves some manual work, in case you are stuck, do contact us. After downloading the graphs, this experiment can be executed via command-line using our provided scripts, as outlined below:

```
$ bash test-igraph-leiden.sh [data-dir] [logs-dir]
```

You can specify the data and logs directories optionally; if not provided, the graphs will be accessed from the "**./Data**" directory, and the output logs will be saved to the "**./Logs**" directory. Expect the experiment to run for approximately 2 days. Once the experiment is complete, the logs will be saved in “***[logs-dir]/igraph--igraph--leiden-communities-icpp2024.log***”. This log file can be converted to a CSV file as follows:

```
$ node process-igraph-leiden.js csv <log-file> <log-file>.csv
```

With the CSV file generated, import it into a new sheet of the puzzlef/leiden-communities-openmp--icpp spreadsheet, and then transfer these numbers (*duration* to *igraph-t*, *modularity* to *igraph-m*, and *disconnected_communities_fraction* to *igraph-d*) to the **“compare”** sheet with column names **“igraph-*”**. This action will update the figures in the "**compare**" sheet, resembling **Figure 6** in our paper.

Measuring Performance of NetworKit Leiden

In **Section 5.2** of our paper, we compare the performance of GVE-Leiden with NetworKit Leiden. Running this experiment involves some manual work, in case you are stuck, do contact us. After downloading the graphs, this experiment can be executed via command-line using our provided scripts, as outlined below:

```
$ bash test-networkit-leiden.sh [data-dir] [logs-dir]
```

You can specify the data and logs directories optionally; if not provided, the graphs will be accessed from the "**./Data**" directory, and the output logs will be saved to the "**./Logs**" directory. Expect the experiment to run for approximately 1 day. Once the experiment is complete, the logs will be saved in "***[logs-dir]/networkit--networkit--leiden-communities-icpp2024.log***". This log file can be converted to a CSV file as follows:

```
$ node process-networkit-leiden.js csv <log-file> <log-file>.csv
```

With the CSV file generated, import it into a new sheet of the puzzlef/leiden-communities-openmp--icpp spreadsheet, and then transfer these numbers (*total_time* to *networkit-t*, *modularity* to *networkit-m*, and *disconnected_communities_fraction* to *networkit-d*) to the **"compare"** sheet with column names **"networkit-*"**. This action will update the figures in the "**compare**" sheet, resembling **Figure 6** in our paper.

Measuring Performance of cuGraph Leiden

In **Section 5.2** of our paper, we compare the performance of GVE-Leiden with cuGraph Leiden. Running this experiment involves some manual work, in case you are stuck, do contact us. If cuGraph is not installed, please install it using the following commands:

```
$ conda create --name cugraph-env -y
$ conda activate cugraph-env
$ conda install -c rapidsai -c conda-forge -c nvidia cugraph
cuda-version=11.4 -y
```

After downloading the graphs and installing **cugraph**, this experiment can be executed via command-line using our provided scripts, as outlined below:

```
$ conda activate cugraph-env
$ bash test-cugraph-leiden.sh [data-dir] [logs-dir]
```

You can specify the data and logs directories optionally; if not provided, the graphs will be accessed from the "**./Data**" directory, and the output logs will be saved to the "**./Logs**" directory. Expect the experiment to run for approximately 1 day. Once the experiment is complete, the logs will be saved in "***[logs-dir]/test-cugraph-leiden--main-icpp2024.log***". This log file can be converted to a CSV file as follows:

```
$ node process-cugraph-leiden.js csv <log-file> <log-file>.csv
```

Once the CSV file is generated, import it into a new sheet in the puzzlef/leiden-communities-openmp--icpp spreadsheet. Next, transfer these numbers (*time* to *cugraph-t*, *modularity* to *cugraph-m*, and *disconnected_communities_fraction* to *cugraph-d*) to the **“compare”** sheet under column names **“cugraph-*”**. This will update the figures in the **"compare"** sheet to nearly match **Figure 6** in our paper.

Strong scaling on large real-world graphs

Finally, in **Section 5.3** of our paper, the strong-scaling behavior of GVE-Leiden on large real-world graphs is investigated. Once the graphs have been downloaded, this experiment can be executed via command-line using the provided scripts, as outlined below:

```
$ bash run-strong-scaling.sh [data-dir] [logs-dir]
```

The experiment may run for approximately 1 day. After completion, the logs will be stored in “**[logs-dir]/leiden-communities-openmp--strong-scaling-icpp2024.log**”. To convert this log file into a CSV file, follow the steps below:

```
$ node process-strong-scaling.js csv <log-file> <log-file>.csv
```

Now that CSV has been generated, it can be imported into the “**data**” sheet of the puzzlef/leiden-communities-openmp--strong-scaling-icpp spreadsheet. Now the figure in the “**all**” sheet will be updated. The figure should be similar to **Figure 9** in our paper.