Bitanes2 - Pthreads Version

Author

- André Bannwart Perina
- Betweenness algorithm developed by Ulrik Brandes
    - Brandes, Ulrik. "A faster algorithm for betweenness centrality." Journal of mathematical sociology 25.2 (2001): 163-177.

Introduction

This is a simple Pthreads implementation of the betweenness centrality algorithm developed by Brandes. It was developed as a coursework for the "Parallel Programming Introduction" module offered at the University of São Paulo - Brazil.

Licence

GNU General Public Licence (see LICENSE file).

Prerequisites

- Usual GNU C compiler and runtime;
    - GNU C Compiler (gcc, tested with version 7.1.1 20170630);
    - GNU C Library (glibc, tested with version 2.25);
- Pthreads library.

How to Download and Execute

1. Clone repo, access folder and download submodules:

```
git clone -b pthreads https://github.com/comododragon/bitanes2.git
cd bitanes2
git submodule update --init --recursive
```

2. Compile using `make` (see ***Description of Compiling Options*** for compiling settings):

```
make bin/bitanes2
```

3. (a) Execute, passing as argument the input graph, e.g.:

```
./bin/bitanes2 data/small/er_20_4_03.net
```

3. (b) By default, Pthreads will use 2 threads + the master thread. You can override this setting by passing the number of threads as argument:

```
./bin/bitanes2 data/small/er_20_4_03.net 8
```

4. The results will be available in the same folder as the input file, with the extension `.btw` (e.g. `data/small/er_20_4_03.btw`)

Description of Compiling Options

The Makefile provided with this project has some compilation options:

```
make bin/bitanes2 DEBUG=yes GPROF=yes GCOV=yes
```

where:

- `DEBUG=yes`: Activate debug flag `-g`;
- `GPROF=yes`: Activate flags for profiling with `gprof`;
- `GCOV=yes`: Activate flags for coverage test with `gcov`;

If one wants to change the options after compiling once, run `make clean` ***first.***

Performance

All executions were performed on a Intel Xeon E5-1607. The sequential version (master branch) of this repository was used as baseline.

- Sequential
  - ***Execution times:***
    - ***Large graph 1 (2000 nodes, 31744 edges, filename data/big/ba_2000_32_01.net):*** 2.535 s
    - ***Large graph 2 (10000 nodes, 159744 edges, filename data/big/ba_10000_32_00.net):*** 61.4 s
- Pthreads (4 threads + master thread)
  - ***Execution times:***
    - ***Large graph 1 (2000 nodes, 31744 edges, filename data/big/ba_2000_32_01.net):*** 0.850 s
    - ***Large graph 2 (10000 nodes, 159744 edges, filename data/big/ba_10000_32_00.net):*** 32.954 s
  - ***Speedup:***
    - ***Large graph 1:*** 2.98x
    - ***Large graph 2:*** 1.86x

File Structure

- `bin`: folder for executable files;
- `data`: dataset of graphs;
  - `big`: large graphs (up to 10000 nodes);
  - `small`: small graphs (up to 20 nodes);
- `include`;
  - `common`;
    - `common.h`: procedures used for error detection and reporting (e.g. assert);
  - `brandes.h`: thread function header;
  - `graph.h`: header of graph data structure;
  - `list.h`: header of list/queue/FIFO data structure;
- `obj`: folder for object files (`.o`);
- `src`:
  - `brandes.c`: thread function;

    - `bitanes2.c`: main function source;
    - `graph.c`: source of graph data structure;
    - `list.c`: source of list/queue/FIFO data structure;
- `Makefile`: project makefile.