NetworKit

NetworKit is a growing open-source toolkit for high-performance network analysis. Its aim is to provide tools for the analysis of large networks in the size range from thousands to billions of edges. For this purpose, it implements efficient graph algorithms, many of them parallel to utilize multicore architectures. These are meant to compute standard measures of network analysis, such as degree sequences, clustering coefficients and centrality (see next sections of this document for a list of existing features and a roadmap of planned features). In this respect, NetworKit is comparable to packages such as NetworkX, albeit with a focus on parallelism and scalability. NetworKit is also a testbed for algorithm engineering and contains a few novel algorithms from recently published research, especially in the area of community detection (see list of publications below).

As of release 2.0, NetworKit is meant to be built into a Python extension module. High-performance algorithms are written in C++ and exposed to Python via Cython. Python in turn gives us the ability to work interactively and a rich environment of tools for data analysis and scientific computing, such as `pandas`, `matplotlib`, `networkx`, `numpy` and `scipy`. A static command line program for community detection can still be built (see build instructions below), but we focus our efforts on the interactive Python environment.

Help

Documentation

In addition to this `Readme`, the `NetworKit_UserGuide` provides an introduction to the NetworKit tools, in the form of an interactive IPython Notebook. The `DevGuide` is meant for developers who would like to contribute. When using NetworKit as a Python module, refer to the docstrings of classes, methods and functions. C++ sources are also documented in Doxygen format.

To convert the documentation markdown files to PDF install the pandoc utility and call the script `docs2pdf.sh`.

E-Mail List

For questions regarding NetworKit, subscribe to our e-mail list (`networkit@ira.uka.de`) and feel free to ask.

Requirements

Compiler:

A C++ compiler supporting C++11 (we use GCC 4.8). The compiler and linker flags `-fopenmp -std=c++11` are required.

Libraries:

- `OpenMP` for parallelism

To avoid possible binary incompatibilities, try to build these libraries with the same compiler that will be used to build NetworKit.

Building the C++ Core only

We recommend SCons for building the C++ part of NetworKit. Individual settings for your environment will be read from a configuration file. As an example, the file `build.conf.example` is provided. Copy this to `build.conf` and edit your environment settings. Then call `scons`.

The call to SCons has the following options:

```
scons --optimize=<level> --target=<target>
```

where <`level`> can be

- `Dbg` debug
- `Opt` optimized
- `Pro` profiling

and <`target`> can be

- `Core` build NetworKit as a library, required by the Python shell
- `Tests` build executable for the unit tests
- `Lib` build NetworKit as a library and create symbolic links

For example, to build NetworKit as an optimized library, run scons –optimize=Opt –target=Core

To speed up the compilation on a multicore machine, you can append `-jX` where X denotes the number of threads to compile with.

Logging is enabled by default. If you want to disable logging functionality, add the following to your scons call:

```
--logging=no
```

Test

You actually don't need to build and run our unit tests. However if you experience any issues with NetworKit, you might want to check, if NetworKit runs properly. Please refer to the `Unit Tests and Testing` section in our DevGuide.pdf

Building NetworKit as a Python Module

As of version 2.0 NetworKit is meant to be used as a Python extension module. To build the module, the following is required:

- Python 3 (>= 3.3 recommended)
- Cython

Additionally, the module uses the following external Python packages:

- networkx
- tabulate
- scipy
- matplotlib

The functionality of NetworKit available to Python is greatly reduced without these packages. These are best installed via `easy_install` or `pip`. (If you have multiple Python installations, be sure you use the commands matching your Python 3 version. In the following examples, `python3` and `pip3` will be used.)

```
pip3 install package_name
```

Then, switch to the top folder and run the script `setup.py` with the following options:

```
python3 setup.py build_ext --inplace [--optimize=V] [-jX]
```

The script will call scons to compile NetworKit as a library and then build the extensions in the folder `src/python`. By default, NetworKit will be built with the amount of available cores in optimized mode. It is possible the add the options `--optimize=V` and `-jN` the same way it can be done to a manual scons call, to specify the optimization level (where `V` is `O`, `D` or `P`) and the number of threads used for compilation. The setup script provides more functionality:

```
python3 setup.py develop [--uninstall] [--optimize=V] [-jX]
```

will compile NetworKit, build the extensions and on top of that temporarily install NetworKit so that it is available on the whole system. This can be undone by adding `--uninstall`.

```
python3 setup.py clean [--optimize=V]
```

will remove the extensions and its build folder as well as call scons to remove the NetworKit library and its build folder specified by `--optimize=V`.

Note: All of the above installation command may require `sudo` privileges depending on your system, so try this accordingly.

Assuming the extension module `NetworKit` exists, it can be imported in python:

```
python3
>>> import NetworKit
```

Interactive Work with NetworKit

With NetworKit as a Python extension module, you get access to native high-performance code and can at the same time work interactively in the Python ecosystem. Although the standard Python interpreter works fine, we recommend IPython as a great environment for scientific computing. IPython can also be installed via `pip` or `easy_install`. For tab completion in ipython you may also need to install `readline`. The following should work: `pip3 install readline ipython`.

After the requirements are satisfied, start IPython and import NetworKit.

```
ipython3
>>> from NetworKit import *
```

Now you should be able to use NetworKit interactively. For usage examples, refer to the `UserGuide`.

IPython Notebook

We recommend that you familiarize yourself with NetworKit through experimenting with the interactive IPython Notebook `NetworKit_UserGuide.ipynb` located in the folder `Doc/Notebooks`. To display and work with these notebooks, you have to start a local notebook server from the terminal with:

```
ipython3 notebook --pylab inline
```

Your default browser should open a web interface named "IPython Dashboard". You can either add `NetworKit_UserGuide.ipynb` from the above mentioned location, or you can point IPython to the location by starting it with

```
ipython3 notebook --pylab inline --notebook-dir=Doc/Notebooks
```

The notebook appears in the list and you can start it by clicking on it.

NetworKit as a library

It is also possible to use NetworKit as a library. Therefore, choose the target `Lib` when compiling NetworKit. The include directives in your C++-application look like the following

```
#include <NetworKit/graph/Graph.h>
```

NetworKit is a symlink to the directory `src/cpp`, so the directory structure from the repository is valid. To compile your application, you need to add the paths for the header files and the location of the library. There is a simple source file to demonstrate this. Feel free to compile `LibDemo.cpp` as follows:

```
g++ -o LibDemo -std=c++11 -I/path/to/repository
    -L/path/to/repository LibDemo.cpp -lNetworKit -fopenmp
```

Contribute

We would like to encourage contributions to the NetworKit source code. See the development guide (`DevGuide.mdown`) for instructions. For support please contact `christian.staudt @ kit.edu`.

Development History

For history and possible future features, refer to the `Roadmap`.

Credits

Responsible Developers

- Christian L. Staudt - `christian.staudt @ kit.edu` - Homepage
- Henning Meyerhenke - `meyerhenke @ kit.edu` - Homepage

Co-Maintainer

- Maximilian Vogel - `maximilian.vogel @ student.kit.edu`

Contributors

- Miriam Beddig
- Stefan Bertsch
- Pratistha Bhattarai
- Andreas Bilke
- Guido Brückner
- Patrick Flick
- Lukas Hartmann
- Daniel Hoske
- Yassine Marrakchi
- Aleksejs Sazonovs
- Florian Weber
- Michael Wegner
- Jörg Weisbarth

External Code

The program source includes: - the *The Lean Mean C++ Option Parser* by Matthias S. Benkmann. - a Python 3 version of the *powerlaw* Python module by Jeff Alstott, Ed Bullmore, Dietmar Plenz

License

Publications

The following is a list of publications on the basis of NetworKit. We ask you to cite the appropriate ones if you found NetworKit useful for your own research.

```
@article{staudt2014networkit,
    Author = {Staudt, Christian L and Sazonovs, Aleksejs and Meyerhenke, Henning},
    Date-Added = {2014-03-13 12:41:29 +0000},
    Date-Modified = {2014-03-13 12:42:47 +0000},
    Journal = {arXiv preprint arXiv:1403.3005},
    Title = {NetworKit: An Interactive Tool Suite for High-Performance Network Analysis},
    Year = {2014}}

@inproceedings{sm2013ehpcdh,
    Author = {Christian L. Staudt and Henning Meyerhenke},
    Booktitle = {proceedings of the 2013 International Conference on Parallel Processing},
    Date-Added = {2013-10-01 08:13:23 +0000},
    Date-Modified = {2013-10-01 08:13:23 +0000},
```

Publisher = {Conference Publishing Services (CPS)},
Title = {Engineering High-Performance Community Detection Heuristics for Massive Graphs},
Year = {2013}}