NetworKit

NetworKit is an open-source tool suite 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. NetworKit is focused on scalability and comprehensiveness. NetworKit is also a testbed for algorithm engineering and contains novel algorithms from recently published research (see list of publications below).

NetworKit is a Python module. High-performance algorithms are written in C++ and exposed to Python via the Cython toolchain. Python in turn gives us the ability to work interactively and a rich environment of tools for data analysis and scientific computing. Furthermore, NetworKit's core can be built and used as a native library if needed.

Installation options

We support three ways to install NetworKit:

- NetworKit Virtual Machine: Download and try NetworKit preinstalled on a virtual machine. This is recommended for users using **Microsoft Windows**.
- Pip install: Download the NetworKit Python package with pip. This is the easier way to get NetworKit but you can only use NetworKit via Python this way.
- Build NetworKit from Source: Clone or download the source code of NetworKit and build the C++ and Python modules from source.

More detailed instructions follow after the requirements section. 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 and jupyterhub as great environments for scientific computing.

Once you have installed NetworKit, please make sure to check out our NetworKit UserGuide for an overview of the features provided in NetworKit.

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, while the documentation for the Python sources can be generated with Sphinx. If you have both utilities installed, the documentation can be easily generated by calling the script `make_docs.sh` in `Doc/docs`.

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

Contact

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

Requirements

You will need the following software to install NetworKit as a python package:

- A modern C++ compiler, e.g.: g++ (>= 4.8) or clang++ (>= 3.7)
- OpenMP for parallelism (usually ships with the compiler)
- Python 3 (>= 3.4 is recommended, 3.3 supported)
- Pip
- installation via pip, we have a script that builds the C++ part of NetworKit, so you can try it without SCons. If you are interested in building different configurations and targets (e.g. unittests) from source, SCons is necessary.
- [for developers: Cython (>= 0.21)]

Installation instructions

Installing NetworKit via pip

NetworKit uses some additional external Python packages. While you do not need them to run NetworKit, it is recommended to install them in order to use all the features of NetworKit:

- scipy
- numpy
- readline
- matplotlib
- networkx
- tabulate

You can use the command `pip3 install scipy numpy readline matplotlib networkx tabulate` on your terminal to install all packages at once. From the list of requirements, you need at least a C++ compiler (including OpenMP) and pip (and optionally SCons). NetworKit can be installed via pip with the following command:

```
[sudo] pip[3] install [--user] networkit
```

During the installation process, the setup will check if the aforementioned external packages are available and print warnings at the end of the installation process.

Note: All of the above installation command may require root privileges depending on your system, so try this accordingly. If you do not have root privileges, add `--user` to your command.

Building NetworKit as a Python Module from source

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 top folder. 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 and the number of threads used for compilation. The setup script provides more functionality and can be used with pip aswell:

```
pip3 install -e .
```

will compile NetworKit, build the extensions and on top of that temporarily install NetworKit so that it is available on the whole system. `pip3 uninstall networkit` will remove networkit.

```
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`.

Interactive environment for working with NetworKit

To check that everything works as expected, open a python terminal and run for example:

```
python3
>>> import networkit
>>> G = networkit.Graph(5)
>>> G.addEdge(0,1)
>>> G.toString()
```

Additionally, we recommend that you familiarize yourself with NetworKit through experimenting with the interactive IPython Notebook `NetworKit_UserGuide.ipynb` located in the folder `Doc/Notebooks`. The user guide also introduces a large portion of NetworKits functionality with usage examples. To display and work with these notebooks, you have to install jupyterhub and start a local notebook server from the terminal with:

```
jupterhub --no-ssl
```

If you run into any problems with jupyterhub, head over to the jupyterhub documentation and make sure, you have the listed packages installed. If the notebook server starts as it is supposed to, your default browser should open a web interface or you have to open it manually. Then you can add `NetworKit_UserGuide.ipynb` from the above mentioned location or browse to the location through the web interface.

To show plots within the notebooks, place the following two lines at the beginning of your notebook:

```
%matplotlib
matplotlib.pyplot as plt
```

Building the C++ Core only

In case you do not need NetworKit's Python functionality, this section describes how to build the C++ parts 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=Lib
```

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
```

Use 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 in the directory `include`is a symlink to the directory `networkit/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. Note, that it is possible to link the different builds (debug, profiling, optimized) 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/include -L/path/to/repository LibDemo.cpp -lNetworKit -f
```

Unit tests

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. The unit tests can only be run from a clone or copy of the repository and not from a pip installation. Please refer to the `Unit Tests and Testing` section in our `DevGuide`.

Known Issues

- Mac OS X 10.10 "Yosemite": Some users have reported compilation problems on Yosemite with g++ 4.9. The compiler errors mention register problems. While the exact reason remains unclear, the actual issue seems to be that the compiler tries to perform a dual architecture build. Fix: Enforce a 64-bit build by prepending `ARCHFLAGS="-arch x86_64"` to your setup/pip command, e.g. as in `sudo ARCHFLAGS="-arch x86_64" python3 setup.py build_ext --inplace -j4` or `sudo ARCHFLAGS="-arch x86_64" pip3 install networkit`.
- NetworKit has not yet been successfully built on **Windows**. This is partially due to the fact that Windows ships without a C++ compiler which is necessary to build the Python extensions. Even with the Visual C++ Redistributable our attempts were not successful. Any help is appreciated. It may be possible to build NetworKit as a library on Windows in environments like MinGW or Cygwin.

Contributions

We would like to encourage contributions to the NetworKit source code. See the development guide (`DevGuide.mdown`) for instructions. For support please contact the mailing list.

Credits

Core Development Team

NetworKit is maintained by the Research Group Parallel Computing of the Institute of Theoretical Informatics at Karlsruhe Institute of Technology (KIT).

Maintainers

- Christian L. Staudt
- Henning Meyerhenke
- Maximilian Vogel

Contributors

- Lukas Barth
- Miriam Beddig
- Elisabetta Bergamini
- Stefan Bertsch
- Pratistha Bhattarai
- Andreas Bilke
- Simon Bischof
- Guido Brückner
- Mark Erb
- Kolja Esders
- Patrick Flick
- Michael Hamann
- Lukas Hartmann
- Daniel Hoske
- Gerd Lindner
- Moritz v. Looz
- Yassine Marrakchi
- Mustafa Özdayi
- Marcel Radermacher
- Klara Reichard
- Matteo Riondato
- Marvin Ritter
- Aleksejs Sazonovs
- Arie Slobbe
- Florian Weber
- Michael Wegner
- Jörg Weisbarth

External Code

The program source includes:

- the *The Lean Mean C++ Option Parser* by Matthias S. Benkmann
- the *TTMath* bignum library by Tomasz Sowa

License

The source code of this program is released under the MIT License. We ask you to cite us if you use this code in your project (c.f. the publications section below and especially the technical report). Feedback is also welcome.

Publications

The NetworKit publications page lists the publications on NetworKit as a toolkit, on algorithms available in NetworKit, and simply using NetworKit. We ask you to cite the appropriate ones if you found NetworKit useful for your own research.